FIXED POINT FORMULATION OF OPTIMALITY CRITERIA FOR EFFICIENT TOPOLOGY OPTIMIZATION

A Dissertation
Presented to
The Academic Faculty

By

Weichen Li

In Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Civil and Environmental Engineering

Georgia Institute of Technology

August 2018

FIXED POINT FORMULATION OF OPTIMALITY CRITERIA FOR EFFICIENT TOPOLOGY OPTIMIZATION

Approved by:

Dr. Phanish Suryanarayana, Advisor
School of Civil and Environmental Engineering
Georgia Institute of Technology

Dr. Glaucio H. Paulino, Advisor
School of Civil and Environmental Engineering
Georgia Institute of Technology

Dr. Arash Yavari
School of Civil and Environmental Engineering
Georgia Institute of Technology

Date Approved: July 23, 2018

Thesis Errata Sheet

Author Weichen Li

Primary Dept. Civil and Environmental Engineering

Degree Master of Science **Graduation date** Aug. 4, 2018

Thesis title

Fixed Point Formulation of Optimality Criteria for Efficient Topology Optimization

Brief description of errata sheet

1. There are a total of 3 corrections that need to be made.
2. The locating pages refer to the page numbers printed on the thesis.
3. The contents to be corrected are highlighted with an underscore.

Number of pages 2 (15 maximum, including this page)

Author: I request that the attached errata sheet be added to my thesis. I have attached two copies prepared as prescribed by the current *Specifications for Thesis Preparation*.

Signature of author [redacted] Date Oct. 15, 2018

Thesis Advisor or **Dept. Chair**: I approve the attached errata sheet and recommend its addition to the student's thesis.

Signature [redacted] Date 10/16/2018

Name Phanish Suryanarayana, Glaucio H. Paulino Thesis supervisor/Dept. Chair

VP for Graduate Education/Faculty Affairs: I approve the attached errata sheet and direct the Institute Archives to insert it into all copies of the student's thesis held by the Georgia Tech Libraries, both print and electronic.

Signature _____ Date _____

Name _____

1. **Page: 6** **Line: 12**

   Original Text: "The tested problems have a density contrast ratio of 10^3 ..."

   Correct Text: "The tested problems have a Young's modulus contrast ratio of 10^3 ..."

2. **Page: 30** **Line: 10**

   Original Text: "PAE parameters: m = 4, q = 4, α = 0.9, β = 4 + loop/50, s = 100"

   Correct Text: "PAE parameters: m = 4, q = 4, α = 0.9, β = 3 + loop/50, s = 100"

3. **Page: 41** **Line: 8**

   Original Text: "Mesh 160×80×80 ... filter radius r = 6, maximum iteration..."

   Correct Text: "Mesh 160×80×80 ... filter radius r = 8 maximum iteration..."

To my parents

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisors, Prof. Phanish Suryanarayana and Prof. Glaucio H. Paulino, for their invaluable guidance and support, which has made this work possible. I have also benefited tremendously in terms of professional knowledge and work attitude through the meetings and discussions with my advisors.

I would also love to thank my dear colleagues, Tuo Zhao, Ke Liu, Heng Chi, Xiaojia Zhang, Oliver Giraldo-Londoño, Jiang Yang, Erol Unal, Larissa Novelino, Emily Sanders, Phanisri Pradeep Pratapa, Javier Vila Moran, Qimen Xu, and Abhiraj Sharma, for their useful advices and discussions on my research. In addition, I hope to give special thanks to my friends, Yijian Zhang, Piyush Sood, Cong Du, and Benedict Wong, for their encouragement and help in many aspects of life.

Most importantly, I hope to thank my parents for their unconditional understanding, support, and love.

Finally, this work is supported by the National Science Foundation (NSF) through the grant CMMI 1663244 (DFT-informed Topology Optimization Upscaling: An Accelerated Fixed-point Iteration Approach).

TABLE OF CONTENTS

Acknowledgments iv

List of Figures viii

List of Symbols and Abbreviations. x

Chapter 1: Introduction 1

1.1 Motivation 1

1.2 Thesis Organization 7

Chapter 2: Fixed-Point Iteration Methods: Background 8

2.1 Fixed-Point Iteration 8

2.2 Simple Mixing 9

2.3 Newton's Method 9

2.4 Quasi-Newton Methods 10

2.4.1 Broyden's Method 11

2.4.2 Anderson Mixing 12

2.5 Periodic Anderson Extrapolation (PAE): a mixed method 14

Chapter 3: Fixed-Point Iteration Accelerated OC Update 15

3.1 Fixed-Point Iteration Accelerated Update 15

3.1.1 Topology Optimization as a Fixed-Point Problem 15

3.1.2 Fixed-Point Iteration Accelerated OC Update 17

3.2 Pros and Cons of Several Fixed-Point Iteration Methods: In The Context of Topology Optimization 18

3.2.1 Simple Mixing 19

3.2.2 Broyden's Method 20

3.2.3 Anderson Mixing 21

3.2.4 Periodic Anderson Extrapolation (PAE) 23

3.3 PAE-OC: Parameter Ranges and Practical Implementation 23

Chapter 4: Numerical Examples 25

4.1 Measure of Performance 25

4.1.1 The Change of Design Variables 25

4.1.2 The Objective Function Value 25

4.1.3 The Total Computational Time 25

4.1.4 The Design Evolution 26

4.2 Numerical Examples: 2D Problems 26

4.2.1 Convergence Speedup in the Objective Function, Residual, and Computation Time 26

4.2.2 Convergence Speedup in the Design Evolution 36

4.3 Numerical Examples: 3D Problems 39

4.4 Discussions 43

Chapter 5: Conclusions 44

References 46

LIST OF FIGURES

1.1 Design Domain and Results of a MBB beam with a concentrated load (Ex. 1) 2

1.2 Design Domain and Results of a MBB beam with a distributed load (Ex. 2) 3

4.1 MBB beam with a concentrated load 26

4.2 Ex1. Design Results and Performance of standard OC and PAE-OC (p=1) . 27

4.3 MBB beam with a distributed load 28

4.4 Ex.2 Design Results and Performance of standard OC and PAE-OC 29

4.5 Ex.3 Design Results and Performance of standard OC and PAE-OC (mesh: 100×50) 31

4.6 Ex.3 Design Results and Performance of standard OC and PAE-OC (mesh: 600×300) 32

4.7 Cantilever beam with a concentrated load 33

4.8 Ex.4 Design Results and Performance of standard OC and PAE-OC 34

4.9 Simply supported beam with a concentrated Load 35

4.10 Ex.5 Design Results and Performance of standard OC and PAE-OC 35

4.11 Design domain and boundary conditions 37

4.12 Ex.6 Design Results and Evolution of standard OC and PAE-OC 38

4.13 3D simply supported beam with a concentrated load 39

4.14 Ex.7 Design Results and Performance of standard OC and PAE-OC 40

4.15 3D cantilever beam with a concentrated load 41

4.16 Ex.8 Design Results and Performance of standard OC and PAE-OC 42

LIST OF SYMBOLS AND ABBREVIATIONS

α	Mixing parameter of simple mixing
a	Lower bound of a scalar variable
$\boldsymbol{A}$	Matrix of a linear system
$\boldsymbol{b}$	Right-hand side vector of a linear system
b	Upper bound of a scalar variable
β	Mixing parameter of Anderson mixing
ϵ	Tolerance for convergence
$\boldsymbol{x}$	Variable of functions
$\boldsymbol{x}^*$	Solution of an equation
$\boldsymbol{x}_k$	Variable of functions or densities at the *k*th step
$\overline{\boldsymbol{x}}_k$	Weighted average of function variables at the previous *m* steps counting from $\boldsymbol{x}_k$
$\widetilde{\boldsymbol{x}}$	Vector mapped or updated by a function
$\widetilde{\boldsymbol{x}}_\mathrm{k}$	$\widetilde{\boldsymbol{x}}$ at the *k*th step
$\Delta\boldsymbol{x}_k$	Difference of variables at 2 consecutive steps: $\boldsymbol{x}_{k+1} - \boldsymbol{x}_k$
$\boldsymbol{X}_k$	Matrix with the columns vectors $\Delta\boldsymbol{x}_{k-m}$, $\Delta\boldsymbol{x}_{k-m+1}$, …, $\Delta\boldsymbol{x}_{k-1}$
$\boldsymbol{f}$	Target equation to be solved
$\boldsymbol{f}_k$	Function value or residual evaluated at $\boldsymbol{x}_k$
$\overline{\boldsymbol{f}}_k$	Weighted average of function values at the previous *m* steps counting from $\boldsymbol{f}_k$
$\Delta\boldsymbol{f}_k$	Difference of function values at 2 consecutive steps: $\boldsymbol{f}_{k+1} - \boldsymbol{f}_k$
$\boldsymbol{F}_k$	Matrix with the columns vectors $\Delta\boldsymbol{f}_{k-m}$, $\Delta\boldsymbol{f}_{k-m+1}$, …, $\Delta\boldsymbol{f}_{k-1}$
$\boldsymbol{g}$	Fixed-point mapping
$\boldsymbol{g}_{update}$	Design update scheme of topology optimization
$\boldsymbol{g}_{OC}$	Standard Optimality Criteria update of topology optimization

$\boldsymbol{G}$	Inverse of the Jacobian matrix of $\boldsymbol{f}$
$\boldsymbol{G}_k$	Approximated inverse of Jacobian matrix of $\boldsymbol{f}$ at the *k*th step
$\boldsymbol{I}$	Identity matrix
$\boldsymbol{J}$	Jacobian matrix of $\boldsymbol{f}$
$\boldsymbol{J}(\boldsymbol{x}_k)$	Jacobian matrix of $\boldsymbol{f}$ evaluated at $\boldsymbol{x}_k$
$\boldsymbol{J}_k$	Approximated Jacobian matrix of $\boldsymbol{f}$ at the *k*th step
$\boldsymbol{K}$	Stiffness matrix
$\boldsymbol{l}$	Load vector
m	Number of histories considered in the Anderson mixing and the PAE method
n	Size of a vector
p	Penalization parameter of the Simplified Isotropic Material with Penalization (SIMP) approach
q	Period of applying the Anderson mixing of the PAE method
r	Filter radius of topology optimization
s	Starting step of PAE acceleration in PAE-OC
$\boldsymbol{u}$	Displacement vector
V	Element-wise summation of $\boldsymbol{x}$ and the volume fraction in topology optimization
ω	Weight of a vector in the Anderson mixing
$\boldsymbol{y}$	Vector of the same size of $\boldsymbol{x}$
$\gamma_i^{(\mathrm{k})}$	Coefficient in the Anderson mixing method, corresponds to the *i*th vector at the *k*th step
$\boldsymbol{\gamma}_{\mathrm{k}}$	Coefficient vector with the terms $\gamma_{k-m}^{(\mathrm{k})}, \gamma_{k-m+1}^{(\mathrm{k})}, \ldots, \gamma_{k-1}^{(\mathrm{k})}$
$\boldsymbol{z}$	General vector
$\|\cdot\|_F$	Frobenius norm of a matrix
$\|\cdot\|_2$	2-norm of a vector
$\|\cdot\|_\infty$	Infinity norm of a vector

AAR	Alternating Anderson-Richardson
BFGS	Broyden-Fletcher-Goldfarb-Shanno method
CG	Conjugate Gradient method
FPA	Fixed-Point iteration Acceleration
GBMMA	Gradient Based Method of Moving Asymptotes
GCMMA	Globally Convergent Method of Moving Asymptotes
GMRES	Generalized Minimal Residual method
MMA	Method of Moving Asymptotes
OC	Optimality Criteria
PAE	Periodic Anderson Extrapolation
PAE-OC	Periodically Anderson Extrapolated Optimality Criteria
PCG	Preconditioned Conjugate Gradient method
PCG IC(0)	Preconditioned Conjugate Gradient method with the zero fill-in, Incomplete Cholesky decomposition as the preconditioner
RMS	Root Mean Square
SIMP	Simplified Isotropic Material with Penalization
SNOPT	Sparse Nonlinear OPTimizer
SQP	Sequential Quadratic Programming

SUMMARY

The traditional Optimality Criteria (OC) update in topology optimization suffers from slow convergence, thereby requiring a large number of iterations to result in only a small improvement in the performance and design. To address this problem, we propose to use a novel fixed-point formulation of the OC update to accelerate the convergence. Such strategies can achieve higher convergence rates without overly complexifying the update process.

In this thesis, we first provide some mathematical background on fixed-point iteration methods. Then, based on theoretical analysis and numerical experiments, we analyze these methods' respective advantages and drawbacks in the context of topology optimization. The analysis focuses on the methods' design update stability, effectiveness in reducing the design cycles, computational cost, and robustness. Through numerical studies, we found one of the methods, called Periodic Anderson Extrapolation (PAE), is the most stable, effective, economic, and robust approach to speed up OC's convergence. The overall update is named Periodically Anderson Extrapolated Optimality Criteria (PAE-OC).

Via several 2D and 3D benchmarks, we demonstrate that the PAE-OC can effectively reduce both the number of iterations and computation time. In addition, this scheme shows good robustness with respect to the change of boundary conditions, problem sizes, and parameters. Finally, we show the scalability of the PAE-OC through a 3D problem consisting of more than 3 million degrees of freedom.

CHAPTER 1
INTRODUCTION

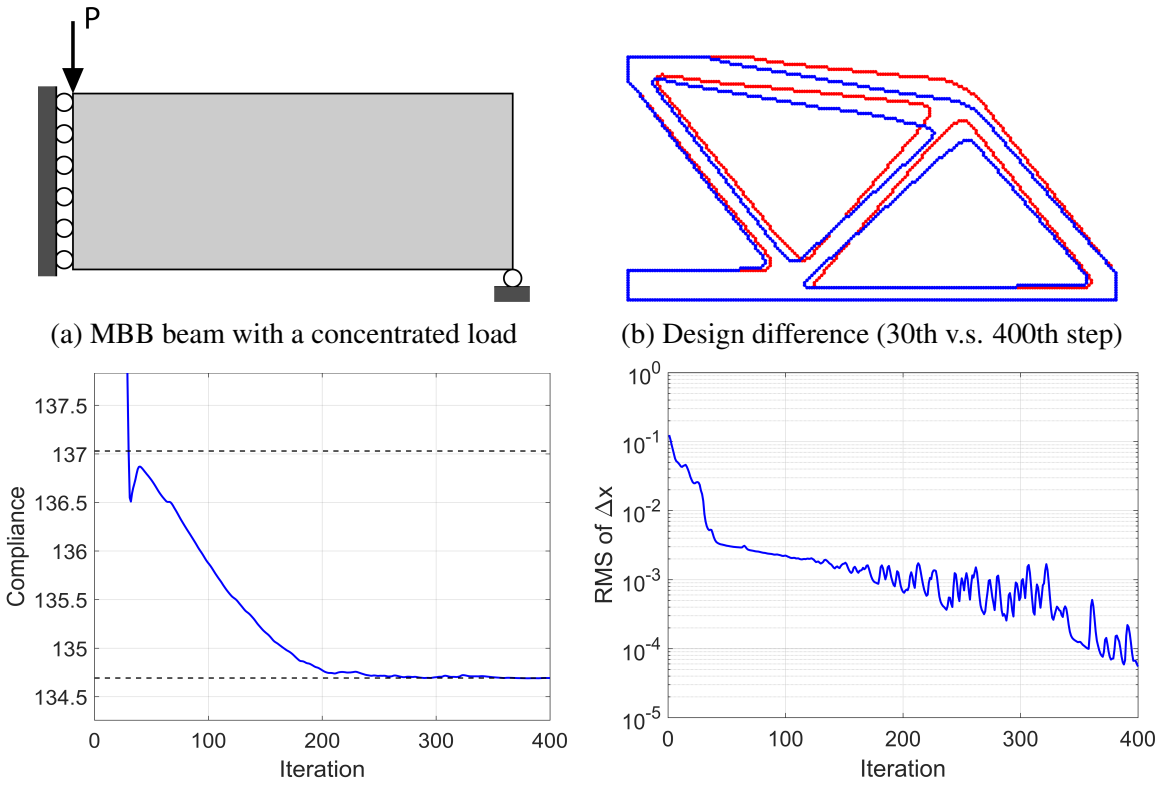
1.1 Motivation

Topology optimization has been used in the design of aeroplanes, vehicles, and even architecture. It allows engineers to obtain designs with optimized performance and minimized costs. Topology optimization can be realized via several approaches, including the density-based method, level set method, topological derivative [1]. Among them, the most popular is the density method with the Simplified Isotropic Mateiral with Penalization (SIMP) approach [2]. Using the SIMP approach, the problem is formulated as a nonlinear programming problem, which is solved by sequential convex programming. Among the most popular convex approximation methods are the first-order Optimality Criteria (OC) method and the Method of Moving Asymptotes (MMA) [3], which use Taylor expansion with respect to some intermediate variables to formulate the convex programming subproblem [4]. The subproblem is solved to update the design variables, which in turn becomes the Taylor expansion point for formulating the next subproblem, and hopefully this iterative procedure converges [4, 5].

In the standard setup, the SIMP approach usually demands hundreds or even thousands of optimization iterations to acquire a converged design, and each iteration requires solving the linear system of equilibrium [6]. Furthermore, the linear system is ill-conditioned due to the high contrast ratio of the densities (e.g. 10^9), which is problematic for the iterative solvers. These features result in the high computation cost of topology optimization, and therefore, reducing the cost remains a challenge in the community [6].

To achieve the cost reduction, various approaches have been proposed. These remedies, according to their targets, can be divided into two categories: 1. those aiming to reduce the

number of design cycles, see, for instance [7, 8, 9, 10, 11, 12], and 2. those aiming to reduce the cost of solving the linear system with iterative solvers, see, for instance [13, 14, 15, 16, 17, 18]. This study belongs to the first type which aims to develop update schemes with high convergence rates. It should be noted that analogous schemes can be used to accelerate the linear solver for large systems in parallel computations [19, 20]. The need for such schemes comes from the fact that the OC or MMA requires a large number of design cycles to achieve an optimized and stable design. Worse still, after the first few steps, the large number of subsequent design cycles result in only a slight improvement of the objective, although the geometric change of the design can be large [1, 4, 5]. Since each iteration requires solving a linear system, the overall process becomes extremely inefficient.



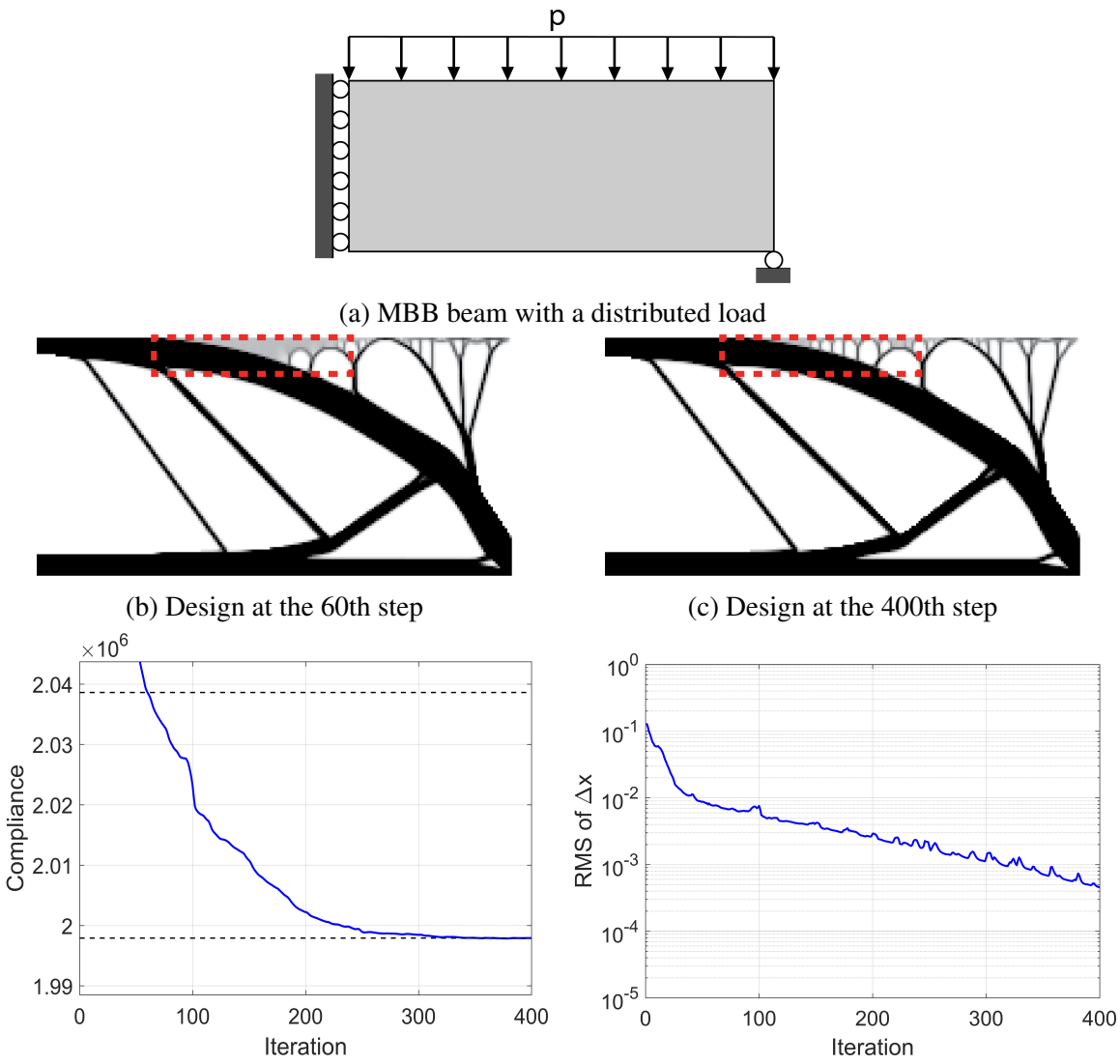
(a) MBB beam with a concentrated load

(b) Design difference (30th v.s. 400th step)

(c) Objective function history

(d) Density change history

Figure 1.1: Design Domain and Results of a MBB beam with a concentrated load (Ex. 1)



(a) MBB beam with a distributed load

(b) Design at the 60th step

(c) Design at the 400th step

(d) Objective function history

(e) Density change history

Figure 1.2: Design Domain and Results of a MBB beam with a distributed load (Ex. 2)

To illustrate the inefficiency, consider the two MBB beam problems shown in Figure 1.1a and Figure 1.2a. The first MBB beam is subjected to a concentrated load, and the second one is subjected to a distributed load. We use the top88 code [21] to solve both problems, with a 200×100 mesh, a penalization parameter of 3, the sensitivity filter, and a maximum number of iterations of 400. We hope to demonstrate the slow convergence from three perspectives: the design evolution, the objective improvement, and the density change.

Example 1: MBB beam with a concentrated load

Figure 1.1b shows the design boundaries by a 0.5 density cut-off at the 30th step (red) and the final 400th step (blue), which indicates that the last 370 iterations only contribute to the slight geometric adjustment. Figure 1.1c is the zoomed-in plot of the objective history, and the 2 dash lines correspond to the values at the 30th step (137.0) and 400th step (134.7). If the 400th step value is set as a standard, the 370 steps lead to only a 2% improvement in the objective. Figure 1.1d shows the root mean square (RMS) of the stepwise density difference. As can be seen, the decline rate after the 30th iteration is much lower than that in the first 30 iterations. To conclude, 92.5% cost results in the small change of the design and the 2% gain in performance.

Example 2: MBB beam with a distributed load

Figure 1.2b and Figure 1.2c show the design at the 60th and 400th iteration. The major difference between the designs is in the red dash boxes where the gray areas are dissipating gradually and evolving to the black and white designs. Figure 1.2d shows the zoomed-in objective history plot, and the 2 dash lines match the value at the 60th step (2.04×10^6) and 400th step (2.00×10^6). Again, the objective improves only 2% at the expense of 340 iterations. Figure 1.2e shows the history of the density change, which drops much more slowly after the 60th step than before. In this example, 85% cost leads to the dissipation of the local gray regions and the 2% improvement of the objective.

The two examples reveal that after the first few iterations, the convergence becomes slow in the geometric change of the design, the improvement of the objective, and the decline of the density change. In the hope of accelerating the convergence, several second-order approaches have been proposed.

Quite a few studies focus on incorporating second-order information with several versions of the MMA, including the original MMA [3], the Global Convergent MMA (GCMMA) [22], and the Gradient Based MMA (GBMMA) [23]. Smaoui et al.[7] and Fleury [8]

showed that the asymptotes of the MMA can be automatically chosen by considering the diagonal elements of the Hessian. Such second order method was tested on benchmark problems including the cantilever beam cross-section optimization problem, the 2-bar truss problem, and the 10-bar truss problem, whose objectives and constraints are explicitly known with respect to the design variables, which makes the exact Hessian computable. The GCMMA2 makes use of the diagonal Hessian elements to determine the coefficients of the GCMMA, but the global convergent property is lost [24]. In the case that the Hessian is not computable or too expensive to evaluate, Duysinx [9] proposed a generalized MMA (GMMA) algorithm that takes into account the second-order information by using a modified BFGS algorithm to update the diagonal approximated Hessian. By means of a topology optimization problem of size 40×26, it is shown to have a higher convergence rate near the optimum than the reciprocal variable method, but whether this approach works for larger problems is unclear. Similarly to GCMMA2, a GBMMA2 method computes the approximated diagonal Hessian by taking the backward difference of the first order derivatives at the two consecutive steps [10]. However, to the best of author's knowledge, these second order MMA methods have yet to be demonstrated to be efficient for large scale topology optimization problems.

Other second-order schemes include the general-purpose sequential quadratic programming (SQP) and the interior point algorithm. Susana et al. [11] proposed a SQP algorithm (TopSQP) for topology optimization, which was tested with 2D problems with a maximum mesh size of 120×60 against other versions of SQP algorithms, interior point algorithms, and the GCMMA. The performance of the methods was evaluated using the performance profile [25]. According to the results, the proposed TopSQP algorithm tends to produce designs with lower objectives and fewer number of iterations than the others [11]. However, it is not as competitive in terms of computational time, indicating the step-wise cost is relatively high. On the other hand, another SQP solver, SNOPT [26], performs the best in both the number of iterations and computational time. Susana et al. [27] benchmarked quite

a few optimization solvers for topology optimization, including the interior point method in Matlab's FMINCON [28], SQP solver SNOPT [26], interior point solver IPOPT [29], OC, MMA, and GCMMA. These methods were tested on the 2D minimum compliance problem, minimum volume problem, and compliant mechanism. The maximum mesh size of the minimum compliance problem is 400×100. The result of the minimum compliance problems shows that the OC and the nested-formulation SQP solver (SNOPT-N) are the most efficient in terms of computational time, although the first-order OC consumes more iterations than the SNOPT. These studies show the SNOPT with nested formulation is competitive in both computational time and the number of iterations, but whether it is scalable for 3D problems remains in question. Susana et al. [12] proposed an interior point algorithm TopIP to solve 3D compliance minimization problems of 1 million elements with iterative linear solvers. The tested problems have a density contrast ratio of 10^3, which is much smaller than the common practice of 10^9 and eases the difficulty of solving the linear system. Nevertheless, the algorithm achieves a relatively small and size-independent number of design cycles. The TopIP costs more time for solving medium-size problems than the first-order GCMMA, and less when the size becomes larger. The main challenge of the interior point method is solving the saddle-point system to obtain the search direction [12]. Despite its relative complexity to first-order methods, the algorithm reveals better scalability compared to GCMMA. However, its effectiveness under higher contrast ratios and its relative performance compared to the classic OC are unknown.

The inefficiency of first-order methods and some limitations of the existing second-order methods motivate the focus of this study: developing an effective, fairly simple, and scalable update approach to speed up the convergence. To this end, we resort to fixed-point iteration techniques to accelerate the convergence on top of the standard OC update. While the second-order methods have been studied quite a bit, the idea of combining fixed-point acceleration techniques with a simple update method (e.g. the OC update) has not been investigated in the topology optimization field. Such strategy has the potential to achieve a

high convergence rate while retaining the OC's simplicity. The latter feature makes room for the scalability that some existing second-order methods lack.

The basic idea of this strategy is to view the optimization update (here, the standard OC update) as a fixed-point problem in which we hope to find the fixed point of the update mapping. In this context, we seize to incorporate fixed-point iteration techniques to develop an efficient and scalable update method.

1.2 Thesis Organization

This thesis is organized as the following. Chapter 2 briefly introduces several fixed-point iteration methods and their mathematical backgrounds. Chapter 3 illustrates how the topology optimization problem can be treated as a fixed-point problem and analyzes the pros and cons of each fixed-point iteration methods in the context of topology optimization. The PAE-OC scheme is also proposed. Chapter 4 demonstrates the convergence speedup of the proposed method via several 2D and 3D numerical examples. Finally, Chapter 5 summarizes the conclusions and proposes directions for further studies.

CHAPTER 2

FIXED-POINT ITERATION METHODS: BACKGROUND

This chapter introduces several well-established fixed-point iteration methods, including the simple mixing, Newton's method, Broyden's method [30], Anderson mixing [31], and the Periodic Anderson Extrapolation (PAE).

2.1 Fixed-Point Iteration

Consider a mapping $\boldsymbol{g} : \boldsymbol{R}^n \to \boldsymbol{R}^n$, $\boldsymbol{x}^*$ is the fixed point of $\boldsymbol{g}$ if $\boldsymbol{x}^* = \boldsymbol{g}(\boldsymbol{x}^*)$. If this format is used as an iteration procedure, i.e. $\boldsymbol{x}_{k+1} = \boldsymbol{g}(\boldsymbol{x}_k)$, then it is called the fixed-point iteration, and hopefully the sequence converges to the fixed point of $\boldsymbol{g}$, i.e. $\boldsymbol{x}_k \to \boldsymbol{x}^*$ as $k \to \infty$. A sufficient condition for convergence is that the $\boldsymbol{g}$ is a contraction mapping within some closed subset of $\boldsymbol{R}^n$. Under this condition, the fixed-point of $\boldsymbol{g}$ is unique [32]. We refer the readers to [32] for the detail and proof of this theorem.

Fixed-point iteration is widely-used to solve nonlinear equations $\boldsymbol{f}(x) = 0$, in which we hope to construct a mapping $\boldsymbol{g}$ whose fixed point is also the solution to $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$, i.e. $\boldsymbol{x}^* = \boldsymbol{g}(\boldsymbol{x}^*)$ and $\boldsymbol{f}(\boldsymbol{x}^*) = \boldsymbol{0}$. Here, we introduce several well-known fixed-point iteration methods.

To avoid potential ambiguity, $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ represents the target equation to be solved, $\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{x})$ denotes the related fixed-point form that has the same root, and $\boldsymbol{f}_k = \boldsymbol{f}(\boldsymbol{x}_k)$ denotes the function value evaluated at $\boldsymbol{x}_k$.

2.2 Simple Mixing

Simple mixing defines the fixed-point iteration as:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha \boldsymbol{f}_k \tag{2.1}$$

where the scalar α is the mixing parameter that can vary from one iteration to another [33].

This scheme has a very simple structure: a linear mix of the variable and function value at the current step. The benefit of this simplicity can be amplified in parallel computing if the function evaluation is a local operation that does not require global inter-processor communications. Despite its simple structure, the method usually consumes many more iterations to converge compared to more sophisticated methods [33], if it converges.

Simple mixing is also known as Richardson iteration in the context of linear systems, where $\boldsymbol{f}_k = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k$ is the residual at the kth step. If the linear system is positive definite, then simple mixing is the same as the steepest descent method with step size α.

2.3 Newton's Method

Newton's method can be viewed as a fixed-point iteration, and it has the following expression:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{J}(\boldsymbol{x}_k)^{-1} \boldsymbol{f}_k \tag{2.2}$$

where $\boldsymbol{J}(\boldsymbol{x}_k)^{-1}$ is the inverse of the Jacobian matrix evaluated at $\boldsymbol{x_k}$.

Newton's method possesses quadratic convergence locally [32], but in many applications, $\boldsymbol{J}(\boldsymbol{x}_k)$ is either too expensive to evaluate or not explicitly computable, and solving $\boldsymbol{J}(\boldsymbol{x}_k)\boldsymbol{z} = \boldsymbol{f}_k$ also increases the cost considerably.

2.4 Quasi-Newton Methods

Quasi-Newton methods avoids computing $\boldsymbol{J}(\boldsymbol{x}_k)$ or solving $\boldsymbol{J}(\boldsymbol{x}_k)\boldsymbol{z} = \boldsymbol{f}_k$ and require only one function evaluation in each nonlinear iterate [33]. Here, we briefly introduce two quasi-Newton methods: Broyden's method and Anderson mixing.

Quasi Newton methods usually have the following form:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{J}_k^{-1}\boldsymbol{f}_k \tag{2.3}$$

or

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{G}_k\boldsymbol{f}_k \tag{2.4}$$

where $\boldsymbol{J}_k$ and $\boldsymbol{G}_k$ are some kind of approximation to the Jacobian matrix and the inverse of the Jacobian matrix at the kth step, respectively.

Standard quasi-Newton methods require the following secant condition [34]:

$$\boldsymbol{J}_{k+1}\boldsymbol{\Delta x}_k = \boldsymbol{\Delta f}_k \tag{2.5}$$

or

$$\boldsymbol{G}_{k+1}\boldsymbol{\Delta f}_k = \boldsymbol{\Delta x}_k \tag{2.6}$$

where $\boldsymbol{\Delta x}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ and $\boldsymbol{\Delta f}_k = \boldsymbol{f}_{k+1} - \boldsymbol{f}_k$.

The above n equations can not uniquely determine $\boldsymbol{J}_{k+1}$ or $\boldsymbol{G}_{k+1}$, which has n^2 unknowns, and different quasi-Newton methods have different ways to construct $\boldsymbol{J}_{k+1}$ or $\boldsymbol{G}_{k+1}$.

2.4.1 Broyden's Method

Broyden's method can be obtained by further imposing the no-change condition [34]:

$$\boldsymbol{J}_{k+1}\boldsymbol{z} = \boldsymbol{J}_k\boldsymbol{z} \quad \forall \boldsymbol{z} \ : \ \boldsymbol{z}^T\boldsymbol{\Delta x}_k = \boldsymbol{0} \tag{2.7}$$

With this condition, the expression can be uniquely determined as [34]:

$$\boldsymbol{J}_{k+1} = \boldsymbol{J}_k + (\boldsymbol{\Delta f}_k - \boldsymbol{J}_k\boldsymbol{\Delta x}_k)\frac{\boldsymbol{\Delta x}_k^T}{\boldsymbol{\Delta x}_k^T\boldsymbol{\Delta x}_k} \tag{2.8}$$

Since the numerator of the second term is an outer product of two vectors, this update is essentially a rank-1 update, and the secant condition (2.5) can be verified by substitution.

From another perspective, it is shown that Broyden's update (2.8) can also be obtained by minimizing $||\boldsymbol{J}_{k+1} - \boldsymbol{J}_k||_F$ with respect to the elements of $\boldsymbol{J}_{k+1}$ [35], which implies Broyden's method (2.8) alters $\boldsymbol{J}_k$ to a "minimal", while satisfying the secant condition.

Knowing (2.3) and (2.5), the expression (2.8) can be written as:

$$\boldsymbol{J}_{k+1} = \boldsymbol{J}_k + \frac{\boldsymbol{f}_{k+1}\boldsymbol{\Delta x}_k^T}{\boldsymbol{\Delta x}_k^T\boldsymbol{\Delta x}_k}$$

In the form of the approximated inverse Jacobian $\boldsymbol{G}_{k+1}$, a similar formula can be obtained by minimizing $||\boldsymbol{G}_{k+1} - \boldsymbol{G}_k||_F$ subjected to (2.6) [34]:

$$\boldsymbol{G}_{k+1} = \boldsymbol{G}_k + (\boldsymbol{\Delta x}_k - \boldsymbol{G}_k\boldsymbol{\Delta f}_k)\frac{\boldsymbol{\Delta f}_k^T}{\boldsymbol{\Delta f}_k^T\boldsymbol{\Delta f}_k} \tag{2.9}$$

Similarly, an equivalent expression is:

$$\boldsymbol{G}_{k+1} = \boldsymbol{G}_k(\boldsymbol{I} - \frac{\boldsymbol{f}_{k+1}\boldsymbol{\Delta f}_k^T}{\boldsymbol{\Delta f}_k^T\boldsymbol{\Delta f}_k})$$

Fang et al. [34] generalized the Broyden's method by imposing the secant condition (2.5) or (2.6) and the no-change condition (2.7) to the previous m consecutive steps instead of just one, and put forward the corresponding rank-m update. We refer the readers to [34] for further studies.

2.4.2 Anderson Mixing

Anderson mixing, which is the same as Pulay mixing and the nonlinear GMRES method [36], makes use of the current and previous steps to extrapolate the next iterate. It is a linear combination of the variables $\boldsymbol{x}$ and function values $\boldsymbol{f}(\boldsymbol{x})$ of the previous m steps [34]. To illustrate the idea, we use the same notation and expression as in [34].

First, define the weighted average of the variables and function values of the previous m steps as:

$$\bar{\boldsymbol{x}}_k = \sum_{j=k-m+1}^{k} w_j \boldsymbol{x}_j = \boldsymbol{x}_k - \sum_{j=k-m}^{k-1} \gamma_j^{(k)} \boldsymbol{\Delta x}_j = \boldsymbol{x}_k - \boldsymbol{X}_k \boldsymbol{\gamma}_k$$

$$\bar{\boldsymbol{f}}_k = \sum_{j=k-m+1}^{k} w_j \boldsymbol{f}_j = \boldsymbol{f}_k - \sum_{j=k-m}^{k-1} \gamma_j^{(k)} \boldsymbol{\Delta f}_j = \boldsymbol{f}_k - \boldsymbol{F}_k \boldsymbol{\gamma}_k$$

where $\boldsymbol{\Delta x}_i = \boldsymbol{x}_{j+1} - \boldsymbol{x}_j$, $\boldsymbol{\Delta} f_j = \boldsymbol{f}_{j+1} - \boldsymbol{f}_j$, $\boldsymbol{X}_k = [\boldsymbol{\Delta x}_{k-m}, ..., \boldsymbol{\Delta x}_{k-1}]$, $\boldsymbol{F}_k = [\boldsymbol{\Delta f}_{k-m}, ..., \boldsymbol{\Delta f}_{k-1}]$, and $\boldsymbol{\gamma}_k = [\gamma_{k-m}^{(k)}, ..., \gamma_{k-1}^{(k)}]$.

The key components to be determined are the weights, or equivalently, the coefficients $\boldsymbol{\gamma}_k = [\gamma_{k-m}^{(k)}, ..., \gamma_{k-1}^{(k)}]$. They are obtained by minimizing $||\bar{\boldsymbol{f}}_k||_2^2 = ||\boldsymbol{f}_k - \boldsymbol{F}_k \boldsymbol{\gamma}_k||_2^2$, which sets the gradient to 0 and results in solving the following linear system:

$$(\boldsymbol{F}_k^T \boldsymbol{F}_k) \boldsymbol{\gamma}_k = \boldsymbol{F}_k^T \boldsymbol{f}_k \tag{2.10}$$

The expression of the Anderson mixing update is:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \bar{\boldsymbol{x}}_k + \beta \bar{\boldsymbol{f}}_k \\ &= \boldsymbol{x}_k + \beta \boldsymbol{f}_k - (\boldsymbol{X}_k + \beta \boldsymbol{F}_k)\boldsymbol{\gamma}_k \\ &= \boldsymbol{x}_k + \beta \boldsymbol{f}_k - (\boldsymbol{X}_k + \beta \boldsymbol{F}_k)(\boldsymbol{F}_k^T \boldsymbol{F}_k)^{-1} \boldsymbol{F}_k^T \boldsymbol{f}_k \end{aligned} \tag{2.11}$$

where the scalar β is the assigned mixing parameter and is usually determined heuristically [36].

When no previous steps are considered, i.e. $m = 0$, the Anderson mixing becomes the simple mixing with β being the mixing parameter. When m is not 0, we need to invert the m by m matrix $\boldsymbol{F}_k^T \boldsymbol{F}_k$, which is often ill-conditioned, but can be computed using the Moore-Penrose pseudoinverse [19].

From the quasi-Newton method's perspective, the approximated inverse Jacobian of the Anderson mixing is:

$$\boldsymbol{G}_{k+1} = -\beta \boldsymbol{I} + (\boldsymbol{X}_k + \beta \boldsymbol{F}_k)(\boldsymbol{F}_k^T \boldsymbol{F}_k)^{-1} \boldsymbol{F}_k^T \tag{2.12}$$

By substitution, it can be shown that (2.12) satisfies the secant condition (2.6) for the previous m consecutive steps, i.e.

$$\boldsymbol{G}_{k+1} \boldsymbol{F}_k = \boldsymbol{X}_k \tag{2.13}$$

Similar to the Broyden's method, the expression for $\boldsymbol{G}_{k+1}$ can also be obtained by minimizing $||\boldsymbol{G}_{k+1} + \beta \boldsymbol{I}||_F$ with respect to the terms of $\boldsymbol{G}_{k+1}$ subjected to (2.13) [34], which indicates that Anderson mixing implicitly formulates an approximated inverse Jacobian that is "closest" to a scaled identity matrix.

In the context of linear systems, it is shown that Anderson mixing with $m = k$, i.e. all histories are considered, is essentially equivalent to the GMRES [37].

2.5 Periodic Anderson Extrapolation (PAE): a mixed method

In large-scale applications, applying quasi-Newton updates in every step can be expensive. One cure to this problem is to apply a quasi-Newton method periodically, and between the two quasi-Newton steps, simple update strategies (e.g. simple mixing) are used.

Here, we introduce the Periodic Anderson Extrapolation method, which is referred to as periodic Pulay [38] in electronic structure calculations and Alternating Anderson-Richardson (AAR) for linear systems [20]. In the latter scenario, it is shown that the AAR has better parallel scalability than the CG and GMRES method [20].

The PAE method applies Anderson mixing every q steps and uses simple mixing for the interval steps. The update scheme has the following expression:

$$\boldsymbol{x}_{k+1} = \begin{cases} \boldsymbol{x}_k + \alpha \boldsymbol{f}_k, & \text{if } k \pmod q \neq 0 \\ \boldsymbol{x}_k + \beta \boldsymbol{f}_k - (\boldsymbol{X}_k + \beta \boldsymbol{F}_k)(\boldsymbol{F}_k^T \boldsymbol{F}_k)^{-1} \boldsymbol{F}_k^T \boldsymbol{f}_k, & \text{if } k \pmod q = 0 \end{cases} \tag{2.14}$$

where q is the preset period of applying the Anderson mixing.

An apparent advantage of this scheme is the reduction in the computation cost due to the simple mixing steps while retaining the higher convergence rate from the Anderson mixing steps to some extent. We refer the readers to [19, 20, 38] for the PAE's performance in solving both nonlinear and linear problems.

CHAPTER 3

FIXED-POINT ITERATION ACCELERATED OC UPDATE

This chapter first illustrates how the optimization update can be treated as a fixed-point problem. Then, via theoretical analysis and numerical studies, we further investigate the pros and cons of the methods introduced in Chapter 2 in the context of topology optimization. Finally, we choose the PAE method to accelerate the standard OC update, which proves to be most robust and efficient.

It should be noted that the OC update is a fixed-point iteration by itself as will be shown in the next section. To avoid potential ambiguity, fixed-point iteration accelerated OC update refers to the strategy that uses the general-purpose fixed-point iteration techniques introduced in Chapter 2 to speed up the convergence of the standard OC.

3.1 Fixed-Point Iteration Accelerated Update

3.1.1 Topology Optimization as a Fixed-Point Problem

As mentioned in Chapter 1, we aim to accelerate the convergence of a simple update method which in general has the form:

$$\boldsymbol{x}_{k+1} = \boldsymbol{g}_{update}(\boldsymbol{x}_k) \tag{3.1}$$

where the $\boldsymbol{g}_{update}$ is the update scheme, for example, the OC.

A local optimum of the optimization problem can be viewed as a fixed point of the $\boldsymbol{g}_{update}$ mapping in that the $\boldsymbol{g}_{update}$ no more alters the input design variable. We construct the equivalent equation $\boldsymbol{f}$ related to this mapping in a simple way:

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{g}_{update}(\boldsymbol{x}) - \boldsymbol{x} \tag{3.2}$$

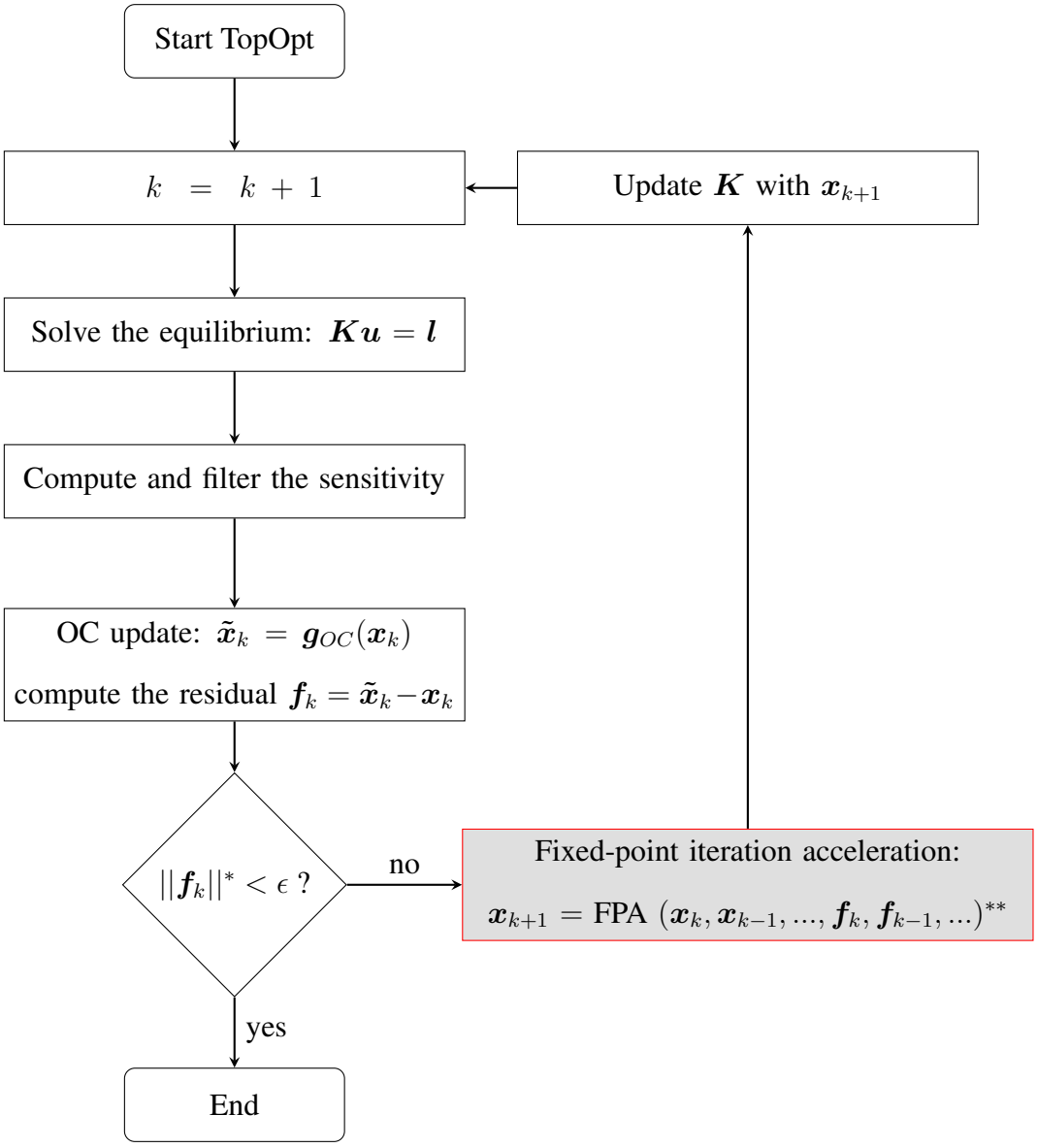
The value of $\boldsymbol{f}$ evaluated at some $\boldsymbol{x}_k$, i.e. $\boldsymbol{f}_k$, can be viewed as the residual of the function at the kth step, and finding a local optimum is equivalent to finding a root of $\boldsymbol{f}$.

Although there are many ways to construct an equivalent equation, this particular form (3.2) is favorable because it requires only the function evaluation and a vector subtraction. In addition, this function is essentially the density change vector of the $\boldsymbol{g}_{update}$, which allows for the same measure of convergence for the $\boldsymbol{g}_{update}$ and the proposed method.

Once the optimization problem is viewed as a fixed-point problem (3.1), all the fixed-point iteration methods become available. Regarding the $\boldsymbol{g}_{update}$, this study adopts the standard OC update (denoted as $\boldsymbol{g}_{OC}$) and focuses on finding the "best" method, among those in Chapter 2, to accelerate the OC.

3.1.2 Fixed-Point Iteration Accelerated OC Update

The topology optimization algorithm with the fixed-point iteration accelerated OC update is shown in the flowchart below.



* This study adopts the root mean square (RMS) of $\boldsymbol{f}_k$ as the measure of convergence.

** For Broyden's method and simple mixing, only the information of last step (i.e. step k) is needed, and for Anderson mixing, the information of the previous m consecutive steps is needed.

As can be seen from the flowchart, the key additional component to the original topology optimization algorithm is the fixed-point iteration step highlighted in the shaded box. In this step, we apply the several fixed-point iteration methods in Chapter 2 to accelerate the convergence. The performance of these methods is evaluated based on the following criteria:

- Convergence robustness with respect to the change of boundary conditions, problem size, and parameters
- Convergence speedup in terms of the design cycle reduction compared to the original OC
- Stepwise computation cost

To test the methods, we use the top88 code [21] as a basis and adopt the default sensitivity filter. The move limits of the OC update is 0.2.

3.2 Pros and Cons of Several Fixed-Point Iteration Methods: In The Context of Topology Optimization

Here, we discuss the pros and cons of the methods introduced in Chapter 2 in the context of topology optimization. These observations are based on both numerical tests and theoretical analyses.

The ideal fixed-point acceleration scheme should be effective in reducing design cycles, robust with respect to parameter changes, inexpensive to compute and store, and scalable in problem size. Additionally, since the design variable is the material density, it is important for the update scheme to preserve the element-wise summation of the variable vector, i.e. the volume.

3.2.1 Simple Mixing

The simple mixing scheme (2.1) essentially updates the densities as the linear mix of the densities at the previous step and the ones updated by the OC at the current step.

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha \boldsymbol{f}_k = (1-\alpha) \cdot \boldsymbol{x}_k + \alpha \cdot \boldsymbol{g}_{OC}(\boldsymbol{x}_k) \tag{3.3}$$

Obviously, when the mixing parameter $\alpha = 1$, the above formula becomes the OC update.

This simple interpolation has an interesting effect on the evolution and convergence of topology optimization. When $\alpha < 1$, the design evolution becomes smoother and more stable than the OC. The latter tends to cause design oscillations. However, a too small α retards the design evolution in that it excessively reduces the step size of the density change. On the other hand, $\alpha > 1$ indicates a linear extrapolation in densities and usually leads to unstable design oscillations and divergence.

Another favorable property of the simple mixing with $\alpha \leq 1$ is the preservation of element-wise summation (volume) and bounds (0 and 1). This property indicates that if $\boldsymbol{x}_k$ and $\boldsymbol{g}_{OC}(\boldsymbol{x}_k)$ satisfy the volume constraint and the density bound of 0 and 1, then the interpolated densities by (3.3) also satisfy these two conditions. The preservation of bounds no more holds when $\alpha > 1$. The proof of both properties are given below.

Proof: preservation of summation (simple mixing)

Let $\tilde{\boldsymbol{x}} = \boldsymbol{g}_{oc}(\boldsymbol{x})$ be the densities updated by OC from $\boldsymbol{x}$, V be the element-wise summation of $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$, and $\boldsymbol{f} = \boldsymbol{g}_{oc}(\boldsymbol{x}) - \boldsymbol{x} = \tilde{\boldsymbol{x}} - \boldsymbol{x}$ be the residual or change of densities. Then we have:

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{x} + \alpha \boldsymbol{f} \\ &= \boldsymbol{x} + \alpha(\tilde{\boldsymbol{x}} - \boldsymbol{x}) \\ &= (1-\alpha)\boldsymbol{x} + \alpha \tilde{\boldsymbol{x}} \end{aligned} \tag{3.4}$$

The element-wise summation is:

$$\begin{aligned}\sum_{j=1}^{n} y_j &= \sum_{j=1}^{n}[(1-\alpha)x_j + \alpha\tilde{x}_j] \\ &= (1-\alpha)\sum_{j=1}^{n} x_j + \alpha\sum_{j=1}^{n}\tilde{x}_j \\ &= (1-\alpha)V + \alpha V \\ &= V\end{aligned} \tag{3.5}$$

Proof: preservation of bounds (simple mixing)

Let $x_j, \tilde{x}_j \in [a,b]\ \forall j$, and $\alpha \in [0,1]$. Since $\alpha \geq 0$ and $(1-\alpha) \geq 0$, the following inequalities holds:

$$\begin{aligned}y_j &= (1-\alpha)x_j + \alpha\tilde{x}_j \leq (1-\alpha)b + \alpha b = b \\ y_j &= (1-\alpha)x_j + \alpha\tilde{x}_j \geq (1-\alpha)a + \alpha a = a\end{aligned} \tag{3.6}$$

Therefore, $y_j \in [a,b]\ \forall j$. Moreover, when $\alpha \in (0,1)$ and $x_j \neq \tilde{x}_j$, we have $\min(x_j, \tilde{x}_j) < y_j < \max(x_j, \tilde{x}_j)$. This partly explains the stabilizing effect in that simple mixing limits the density change caused by the OC update.

Despite the smoothing and stabilizing effect, the simple mixing does not provide greater fundamental benefit for the convergence.

3.2.2 Broyden's Method

As mentioned in Chapter 2, Broyden's method belongs to quasi-Newton methods and is locally superlinearly convergent [32]. Indeed, the convergence rate is higher near the optimum, provided the initial guess of the approximated inverse Jacobian $\boldsymbol{G}_0$ is carefully chosen, and the problem size is small (e.g. 50×25). However, numerical tests showed many disadvantages of applying Broyden's method to the OC update.

First, the direct use of the Broyden's formulas (2.8) or (2.9) often results in dense ap-

proximated Jacobian ($\boldsymbol{J}_k$) or inverse Jacobian matrices ($\boldsymbol{G}_k$), which increases the computation cost due to the step-wise matrix-vector multiplications and demands a much larger memory when the problem size grows.

Second, the initial guess of the approximated inverse Jacobian $\boldsymbol{G}_0$ has a tremendous impact on the stability of the design evolution. Heuristic guesses, for example, a scaled identity matrix, can lead to instability and divergence.

Third, Broyden's method (2.8) or (2.9) in general does not preserves the volume or the density bound, which exacerbates the stability issue. In many tests, the density change of some elements were too large and exceeded the bounds by a large distance.

Finally, among the converged tests, Broyden's method did not lead to a substantial decrease in design cycles, and the step-wise time is much more than the OC. Therefore, Broyden's method is not adopted.

3.2.3 Anderson Mixing

Anderson mixing essentially extrapolates the densities based on the densities $\boldsymbol{x}$ and the change of densities $\boldsymbol{f}$ from the OC update at the previous m steps. It reveals much better stability than Broyden's method, provided the mixing parameters in (2.14) α and β are chosen in certain ranges. The outstanding stability partly comes from Anderson mixing's preservation of element-wise summation, which will be proved below. Anderson mixing is also less sensitive to boundary conditions, problem size, and load cases compared to the Broyden's method.

Proof: preservation of summation

Anderson mixing is the linear combination of the variables and residuals of the previous m steps. Suppose the element summation is V for $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ for the previous m steps, we have:

$$\begin{aligned}
\boldsymbol{y} &= \sum_{i=1}^{m} w_i \boldsymbol{x}_i + \beta \sum_{i=1}^{m} w_i \boldsymbol{f}_i \\
&= \sum_{i=1}^{m} w_i \boldsymbol{x}_i + \beta \sum_{i=1}^{m} w_i (\tilde{\boldsymbol{x}}_i - \boldsymbol{x}_{\boldsymbol{i}}) \\
&= (1-\beta) \sum_{i=1}^{m} w_i \boldsymbol{x}_i + \beta \sum_{i=1}^{m} w_i \tilde{\boldsymbol{x}}_i
\end{aligned} \tag{3.7}$$

The element-wise summation is:

$$\begin{aligned}
\sum_{j=1}^{n} y_j &= (1-\beta) \sum_{j=1}^{n} \sum_{i=1}^{m} w_i x_{i,j} + \beta \sum_{j=1}^{n} \sum_{i=1}^{m} w_i \tilde{x}_{i,j} \\
&= (1-\beta) \sum_{i=1}^{m} w_i \sum_{j=1}^{n} x_{i,j} + \beta \sum_{i=1}^{m} w_i \sum_{j=1}^{n} \tilde{x}_{i,j} \\
&= (1-\beta) \sum_{i=1}^{m} w_i V + \beta \sum_{i=1}^{m} w_i V \\
&= (1-\beta) V + \beta V \\
&= V
\end{aligned} \tag{3.8}$$

Notice that unlike the simple mixing, Anderson mixing does not preserve the bounds of the variables because the equivalent weights w_i computed by the coefficients obtained in (2.10) are not nonnegative in general. Therefore, the bounding inequalities similar to (3.6) do not hold. This may lead to some densities outside the [0,1] bound, and they need to be artificially truncated to match the bounds. However, our numerical study shows that as long as the mixing parameter β is not too big, the small violation of volume due to the density truncation will fade out as the density change $\boldsymbol{f}$ declines. Such small violation of bounds can be further constrained by the PAE method discussed in the next section.

Regarding the speedup, in many cases, Anderson mixing can effectively reduce the number of iterations when the parameters are chosen in certain ranges. Moreover, the step-wise cost is also much cheaper than the Broyden's method because the main additional cost lies in only computing and inverting the $\boldsymbol{F}_k^T \boldsymbol{F}_k$ matrix in (2.11), which is typically small.

Another feature of the Anderson mixing accelerated OC, compared to the original OC, is the increase in storage caused by keeping the information of the previous m steps. Fortunately, in the scope of topology optimization, m is typically small ($3 \sim 5$).

3.2.4 Periodic Anderson Extrapolation (PAE)

PAE applies Anderson mixing periodically and uses simple mixing between the Anderson steps, which further reduces the cost. More importantly, PAE shows better stability with respect to the change of parameters than Anderson mixing. This improvement may relate to the variable bound preservation of the simple mixing in that simple mixing constraints the small volume violation caused by Anderson mixing. In terms of reducing design cycles, PAE also outperforms Anderson mixing in most cases.

To conclude, PAE performs the best in terms of stability in design evolution, robustness with respect to parameter changes, reduction in design cycles, and step-wise cost. Therefore, PAE is adopted to accelerate the standard OC's convergence. The overall update is named Periodically Anderson Extrapolated Optimality Criteria (PAE-OC).

3.3 PAE-OC: Parameter Ranges and Practical Implementation

Parameter values affect the performance of the PAE-OC scheme. These parameters include the number of histories m, the period of applying Anderson mixing q, the mixing parameters of the simple mixing α and Anderson mixing β. Although PAE shows good robustness to parameter changes, numerical tests show that the optimum range is relatively small and insensitive to the change of boundary conditions and problem sizes. The recommended parameter ranges are:

The number of histories: $m = 3 \sim 5$

The period of applying Anderson mixing: $q = 3 \sim 5$

The mixing parameters of simple mixing: $\alpha = 0.8 \sim 0.95$

The mixing parameters of Anderson mixing: $\beta = 2 \sim 9$

The numerical tests show that a too large m retards the design evolution, and a $m < 3$ tends to cause design oscillations. The mixing parameter of the simple mixing step α (smaller than 1) also affects the evolution speed as mentioned in 3.2.1. A too small α restrains the change of the design. Therefore, the suggested range of α can stabilize the evolution without slowing it down. The mixing parameter of Anderson mixing β has a relatively smaller impact on the convergence speedup. In general, a larger β tends to accelerate the evolution, but a too large β can lead to instability, which may be due to the amplification of the volume violation. In practice, β is used in a continuous fashion in which it increases as the iteration proceeds so that it can compensate for the gradual slowing-down of the evolution.

The other practical issue is when to activate the PAE acceleration, i.e. the starting step s. Since our goal, as mentioned in Chapter 1, is to address the slow-down in convergence after the first few steps, the PAE acceleration is applied after the first few iterations ($20 \sim 100$) when the basic topology has been formed. Furthermore, if the PAE starts too early, the evolution may stagnate in that it fails to change to a better design after many steps. This observation complies with the fact that quasi-Newton's superlinear convergence is only local.

CHAPTER 4
NUMERICAL EXAMPLES

This chapter demonstrates PAE-OC's speedup in convergence with several 2D and 3D problems. The convergence speed is evaluated from four perspectives: the drop of the density change, the reduction of the objective function, the total computational time, and the design evolution. Regarding the code, we use the top88 code for 2D problems, and the top3D code [39] for 3D problems. Both codes use the sensitivity filter with move limits 0.2.

4.1 Measure of Performance

4.1.1 The Change of Design Variables

We adopt the RMS of the residual vector $\boldsymbol{f}$ (3.2) (the density change due to the OC update) as the measure of the residual instead of the commonly-used $||\boldsymbol{f}||_\infty$. The RMS value can better reveal the overall change of the design than the $||\boldsymbol{f}||_\infty$. This study uses a 10^{-6} RMS tolerance to terminate the optimization for all problems.

4.1.2 The Objective Function Value

The objective function is the compliance, whose reduction rate is an important qualitative measure for convergence speed. However, due to the non-convex nature of the topology optimization problems, different update schemes may end up in different local optima, and therefore the objective function serves as only an auxiliary measure.

4.1.3 The Total Computational Time

The computational time becomes a practical issue when solving large-scale problems or many medium-scale problems. Here, the computational time is defined as the total time

of the optimization cycles, including solving the linear system, computing and filtering the sensitivity, OC update, and the PAE acceleration. The time excludes preprocessing before entering the optimization loop, plotting updated designs, printing information, and post processing after the optimization loop.

4.1.4 The Design Evolution

The design evolution or the geometric change is depicted with the dissipation of the gray areas. The dissipation rate serves as a qualitative measure of how fast the design is evolving.

4.2 Numerical Examples: 2D Problems

4.2.1 Convergence Speedup in the Objective Function, Residual, and Computation Time

Ex.1 MBB beam with concentrated load (penalization = 1)

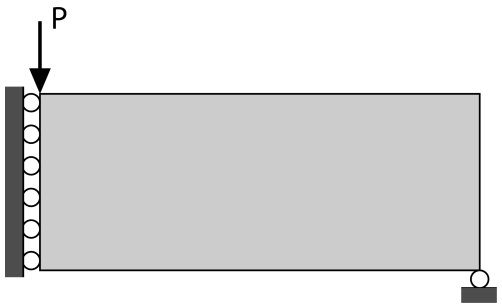
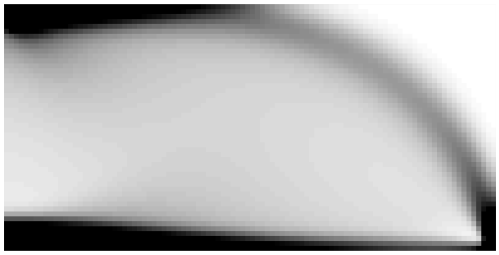


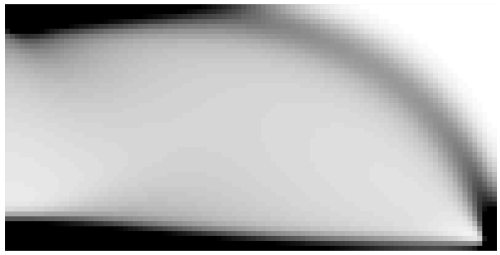
Figure 4.1: MBB beam with a concentrated load

The design domain and boundary conditions are given in Figure 4.1. We test this problem with a penalization parameter equal to 1, which makes the problem convex. Other parameters are given below.

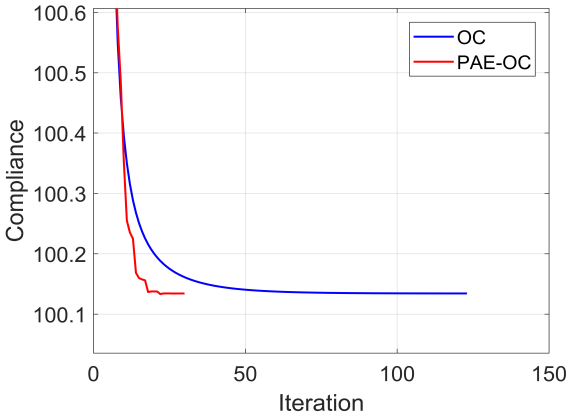
- Mesh 100×50, volume fraction $V = 0.3$, penalization $p = 1$, filter radius $r = 2$, maximum iteration = 150
- PAE parameters: $m = 4$, $q = 4$, $\alpha = 0.9$, $\beta = 3 + loop/50$, $s = 5$



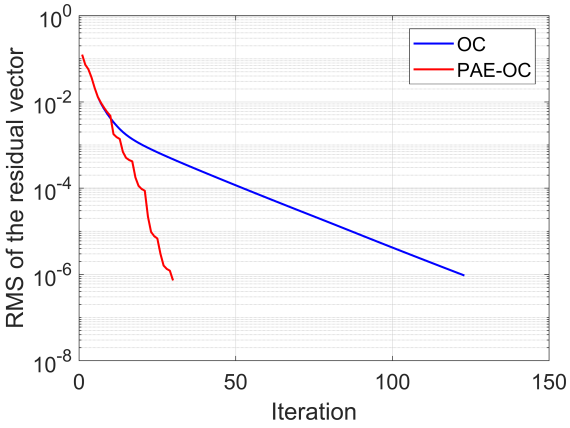
(a) Final design by OC



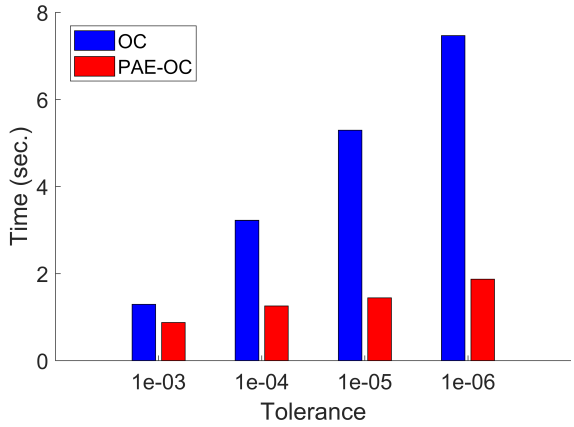
(b) Final design by PAE-OC



(c) Objective function history (p=1)



(d) Residual history



(e) Consumed time

Figure 4.2: Ex1. Design Results and Performance of standard OC and PAE-OC (p=1)

Figure 4.2 shows the results of the standard OC and PAE-OC update. As shown in the objective function plot (Figure 4.2c) and the residual plot (Figure 4.2d), the PAE-OC declines at a much higher rate in both the objective function and the residual than the OC. Figure 4.2e shows the time for the residual to reach various levels of tolerance. The PAE-OC consumes approximately one fourth as much time as the OC.

Ex.2 MBB beam with distributed load (penalization = 1)

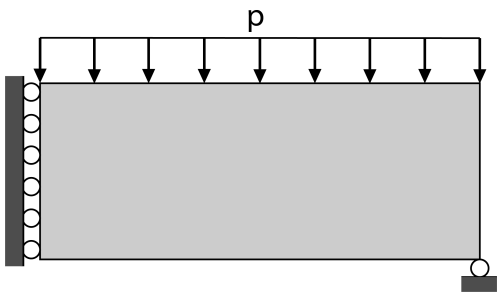
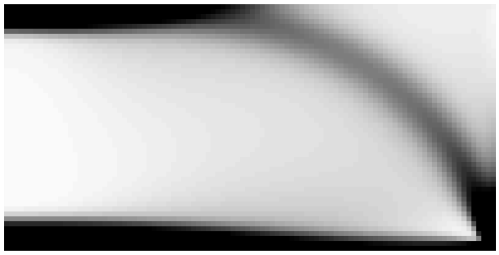
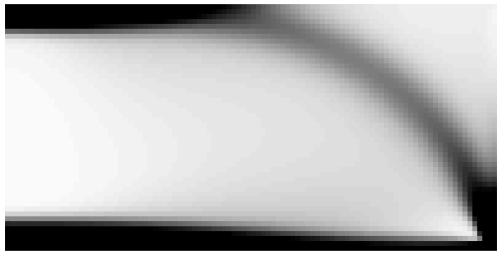


Figure 4.3: MBB beam with a distributed load

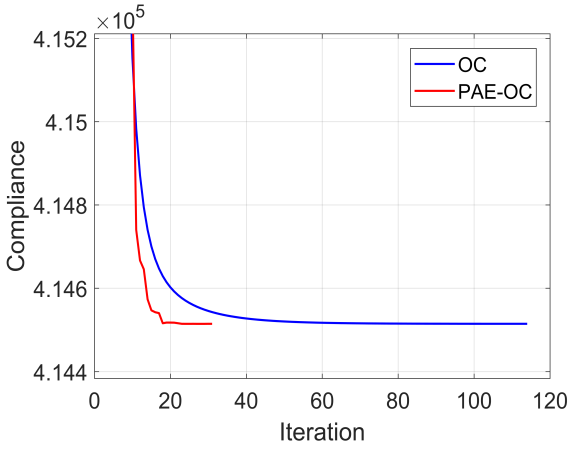
This example changes the loading boundary condition to the distributed load as shown in Figure 4.3, and all other parameters remain the same as Ex.1.



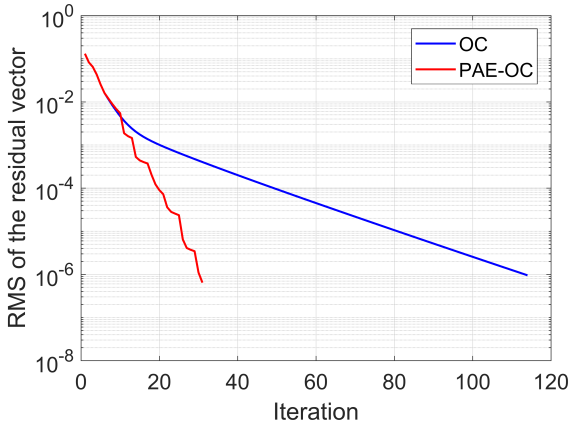
(a) Final design by OC



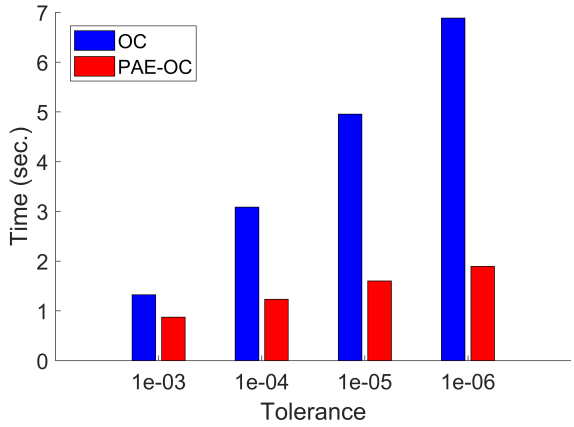
(b) Final design by PAE-OC



(c) Objective function history (p=1)



(d) Residual history (p=1)



(e) Consumed time

Figure 4.4: Ex.2 Design Results and Performance of standard OC and PAE-OC

Figure 4.4 shows the results of this example. Similar conclusions can be drawn: the PAE-OC converges much more efficiently than the standard OC update.

Ex.3 MBB beam with concentrated load and penalization 3

While a penalization parameter greater than 1 drives the design to a black and white structure, it also makes the optimization problem nonconvex. In this example, we examine the PAE-OC's effectiveness with the same MBB beam problem under a penalization $p = 3$. Moreover, the problem is tested with 2 meshes: 100×50 and 600×300, which can reveal PAE-OC's sensitivity to problem sizes. The parameters are given below.

For the smaller problem:

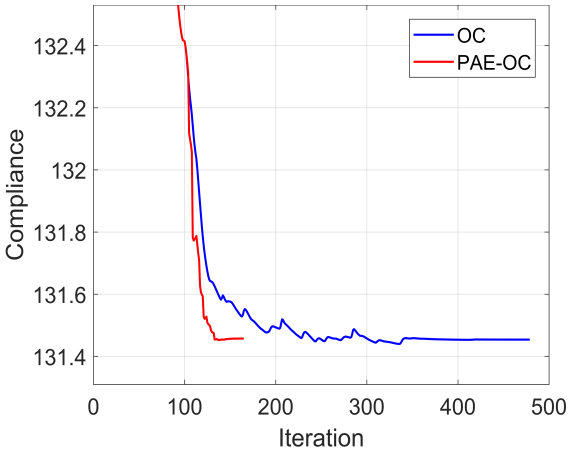
- mesh 100×50, volume fraction $V = 0.3$, penalization $p = 3$, filter radius $r = 2$, maximum iteration = 600
- PAE parameters: $m = 4$, $q = 4$, $\alpha = 0.9$, $\beta = 4 + loop/50$, $s = 100$

For the bigger problem:

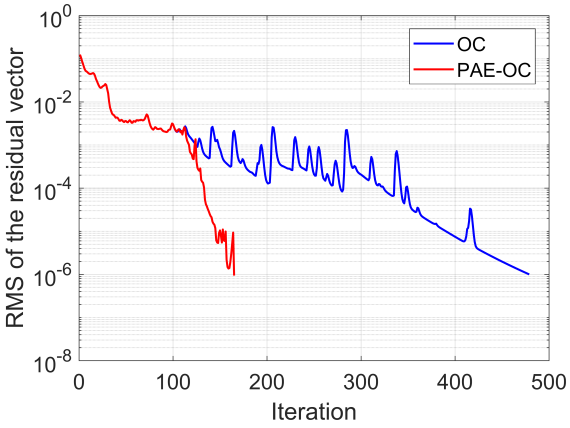
- mesh 600×300, volume fraction $V = 0.3$, penalization $p = 3$, filter radius $r = 12$, maximum iteration = 600
- PAE parameters: $m = 4$, $q = 4$, $\alpha = 0.9$, $\beta = 4 + loop/50$, $s = 50$

(a) Final design by OC

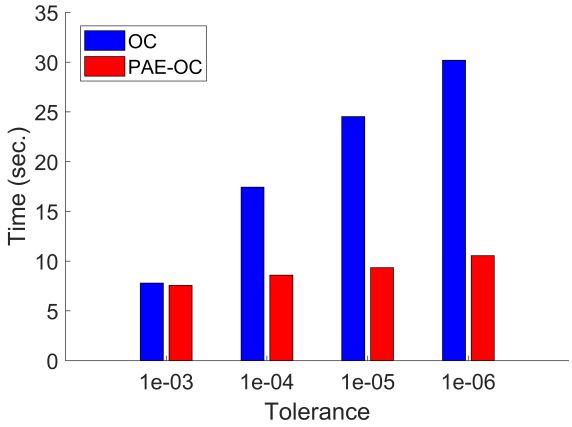
(b) Final design by PAE-OC



(c) Objective function history



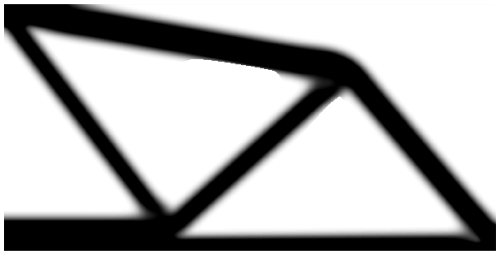
(d) Residual history



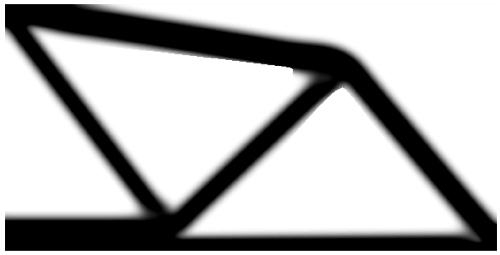
(e) Consumed time

Figure 4.5: Ex.3 Design Results and Performance of standard OC and PAE-OC (mesh: 100×50)

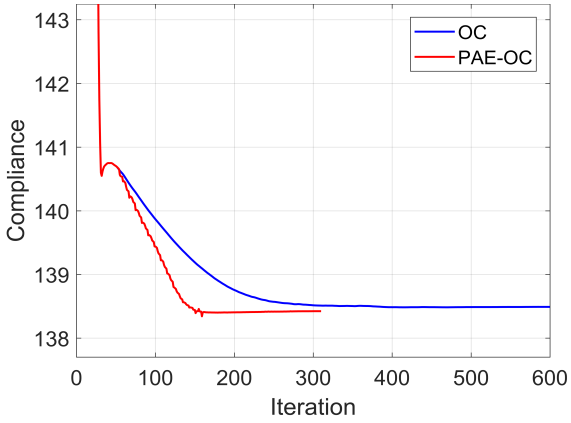
Figure 4.5 shows the results of the smaller problem. Obviously, when the penalization is raised to 3, the convergence of both the objective function (Figure 4.5c) and the residual (Figure 4.5d) becomes less smooth. In this case, the PAE-OC still converges much faster than the OC, consuming roughly one third as many iterations as the OC.



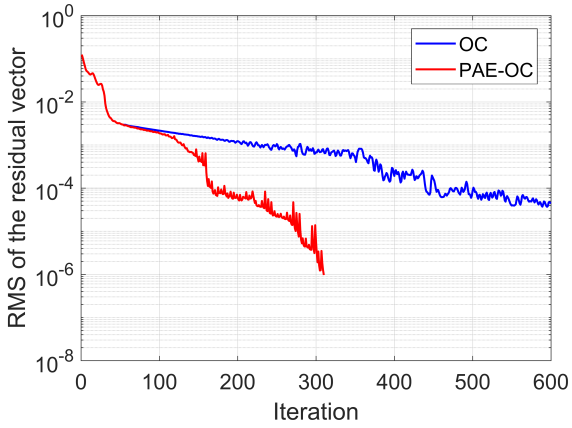
(a) Final design by OC



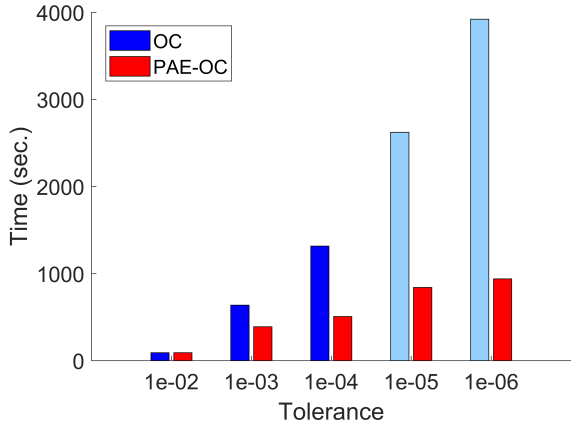
(b) Final design by PAE-OC



(c) Objective function history



(d) Residual history



(e) Consumed time (lighter blue indicates estimated value)

Figure 4.6: Ex.3 Design Results and Performance of standard OC and PAE-OC (mesh: 600×300)

Figure 4.6 shows the results of the bigger problem. Since the residual of the standard OC did not reach 10^{-5} in 600 steps, the OC's time at 10^{-5} and 10^{-6} (light blue bars in Figure 4.6e) are estimated by the linear fitting of residual curve and the average step time. As can be seen, PAE-OC is much more efficient in reducing the objective and the resid-

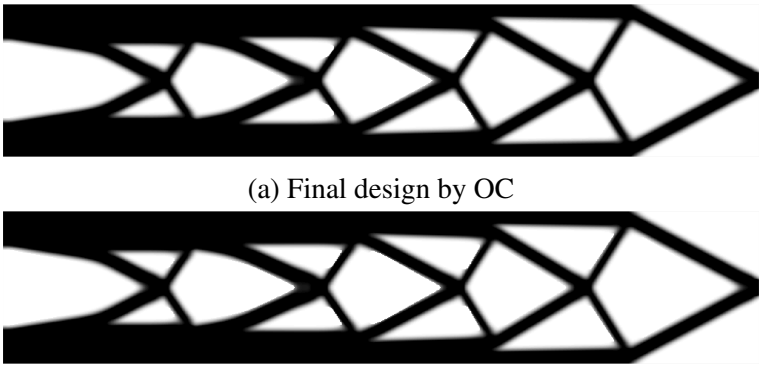
ual. This example also shows that PAE-OC's performance is insensitive to problem sizes. Although the number of iterations rises for both methods as the size increases, PAE-OC grows only 150 in the number of iterations compared to OC's 450 (projected).

Ex.4 Cantilever beam with concentrated load

Figure 4.7: Cantilever beam with a concentrated load

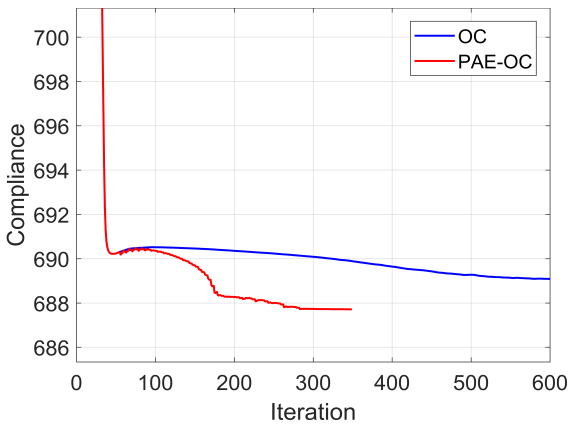
Figure 4.7 shows the design domain and boundary condition of this problem, and the parameters are given below.

- mesh 1000×200, volume fraction $V = 0.5$, penalization $p = 3$, filter radius $r = 8$, maximum iteration = 600
- AAR parameters: $m = 4$, $q = 4$, $\alpha = 0.9$, $\beta = 6 + loop/50 \leq 9$, $s = 50$

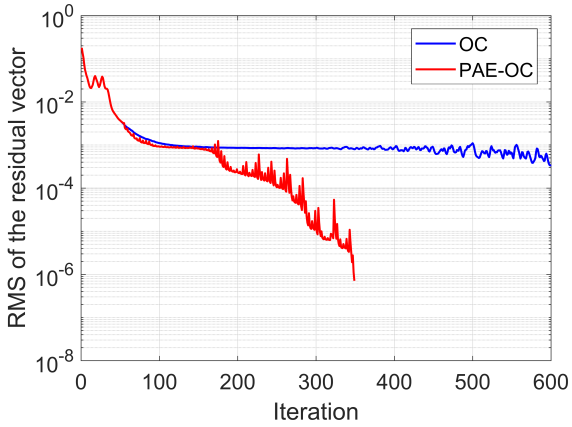


(a) Final design by OC

(b) Final design by PAE-OC



(c) Objective function history



(d) Residual history

Figure 4.8: Ex.4 Design Results and Performance of standard OC and PAE-OC

Figure 4.8 shows the results of the cantilever beam problem. Since the RMS of the residual by standard OC flattens before dropping to 10^{-4}, the time comparison is not plotted.

Similarly, PAE-OC outperforms standard OC from all perspectives. Figure 4.8d illustrates the slow convergence of the standard OC in that the residual curve (blue) becomes almost flat for a large portion of iterations that did not effectively improve the performance. On the other hand, the objective and residual of PAE-OC decline much more efficiently.

Ex.5 The wheel problem

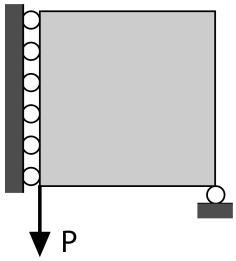
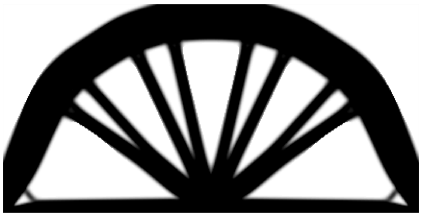


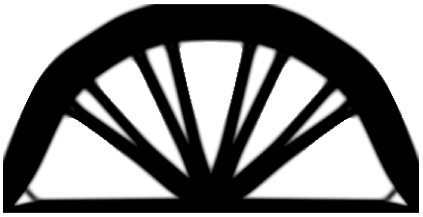
Figure 4.9: Simply supported beam with a concentrated Load

Figure 4.9 shows the design domain and boundary conditions of this problem, and the parameters are shown below.

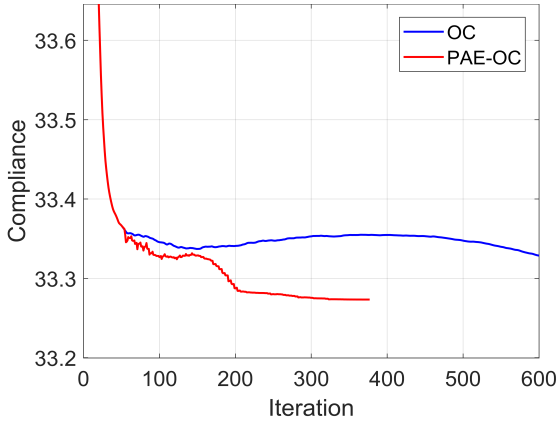
- mesh 200×200, volume fraction $V = 0.5$, penalization $p = 3$, filter radius $r = 4$, maximum iteration = 600
- AAR parameters: $m = 4$, $q = 4$, $\alpha = 0.9$, $\beta = 6 + loop/50 \leq 10$, $s = 50$



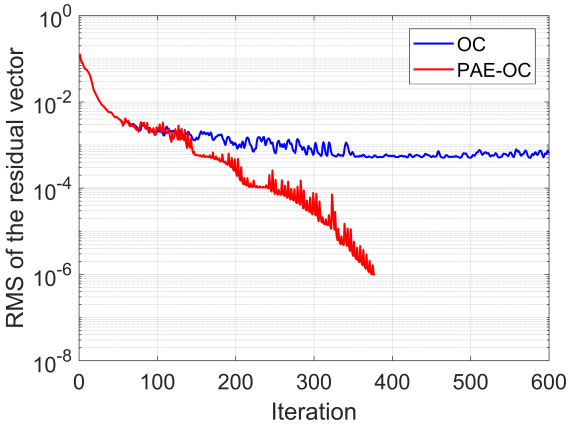
(a) Final design by OC



(b) Final design by PAE-OC



(c) Objective function history



(d) Residual history

Figure 4.10: Ex.5 Design Results and Performance of standard OC and PAE-OC

Figure 4.10 shows the results of this problem. As shown in Figure 4.10d, the residual curve of the standard OC fails to reach 10^{-4} within 600 iterations, and therefore the total time comparison is not plotted. Again, PAE-OC converges much faster than the OC. Although the final results look similar, they are different in the sizes of the triangular holes.

Summary

In this subsection, we demonstrate the PAE-OC's speedup of convergence in terms of the decline of the objective and residual via several benchmark problems. It is shown that PAE-OC can effectively reduce the number of iterations and the computation time. Moreover, PAE-OC is less sensitive to the change of problem sizes than the standard OC in that the number of iterations increases much more slowly as the problem size grows.

4.2.2 Convergence Speedup in the Design Evolution

While the final designs of the two approaches are visually similar, their convergence speed to the black and white solution are different. Here, we illustrate PAE-OC's speedup of the design evolution in terms of the dissipation of the slowly-varying gray regions. To this end, we use the MBB beam example with a distributed load on the top (Figure 4.3), which generates a large area of gray regions. These regions are expected to converge to black and white micro structures.

Ex.6 MBB beam with a distributed load

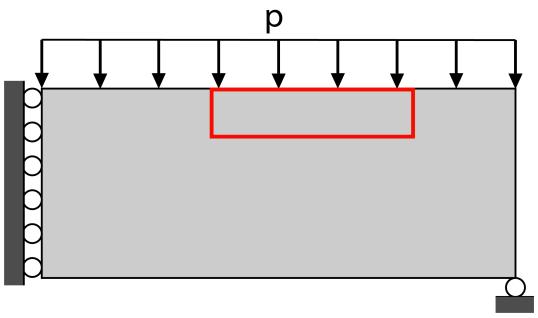
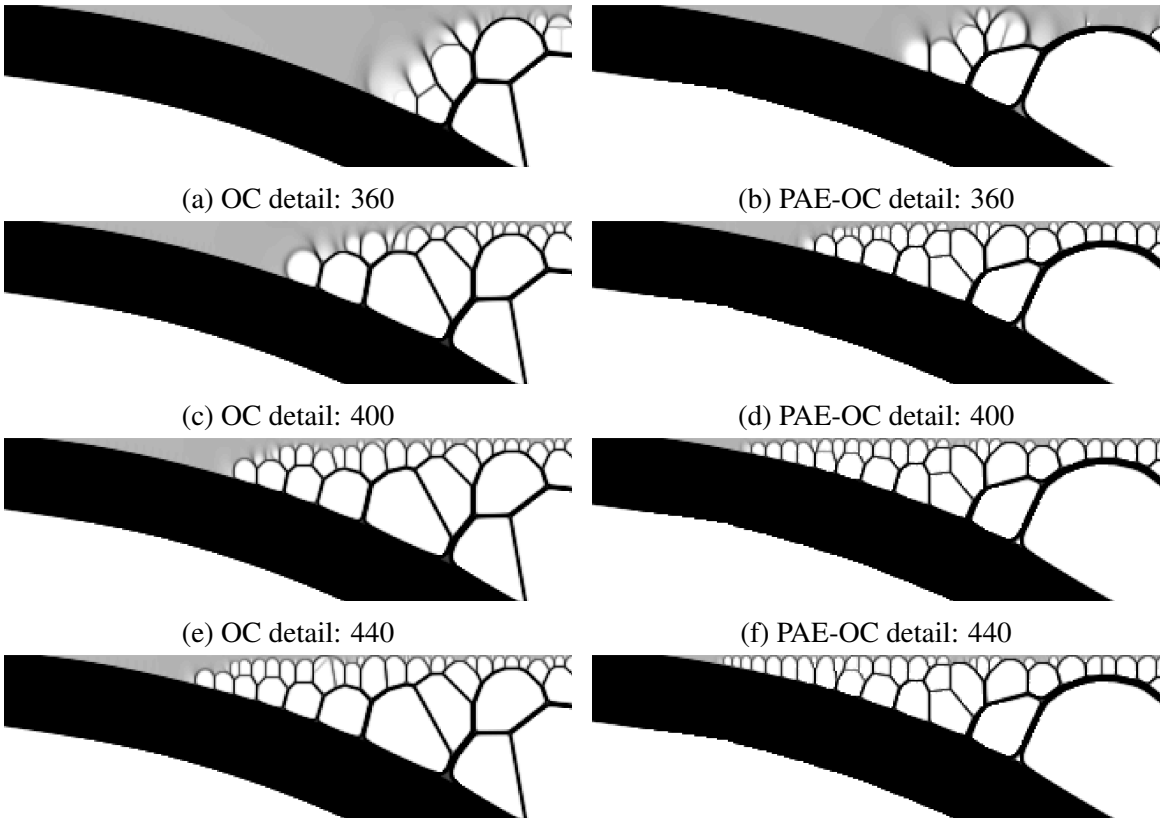


Figure 4.11: Design domain and boundary conditions

The load and boundary conditions are shown in Figure 4.11. This problem will generate many details in the area highlighted by the red box, and we will show the dissipation of gray areas in this box. In order to obtain as many details as possible, we adopt the continuation of the penalization parameter. The penalization changes from 1 to 1.5 at the 50th step, and increases 0.5 per 100 steps till it reaches 3 at the 350 step.

In the case of PAE-OC, we still use the standard OC update for the 10 steps following each penalization change, after which the AAR acceleration is imposed. This can enhance the stability during the few steps after the penalization change. The parameters are listed below.

- mesh 800×400, volume fraction $V = 0.3$, penalization $p = 1 \sim 3$, filter radius $r = 1.2$, maximum iteration = 500
- AAR parameters: $m = 4$, $q = 4$, $\alpha = 0.9$, $\beta = 4 + loop/50 \leq 9$, $s = 10$



(a) OC detail: 360 (b) PAE-OC detail: 360

(c) OC detail: 400 (d) PAE-OC detail: 400

(e) OC detail: 440 (f) PAE-OC detail: 440

(g) OC detail: 480 (h) PAE-OC detail: 480

Figure 4.12: Ex.6 Design Results and Evolution of standard OC and PAE-OC

Figure 4.12 shows the design evolution of the regions in the red box after the penalization turns 3. Clearly, PAE-OC is more efficient and effective in transforming the blurry gray area into sharp black and white details than the OC. The gray area at the 400th step of the PAE-OC is even slightly smaller than that of the standard OC at the 480th step. Also, the two methods ended up in different local optima as can be seen from the different micro structures.

4.3 Numerical Examples: 3D Problems

In this section, we test the PAE-OC with two 3-D examples. We use the Top3d code [39] as a basis, and add the AAR routine as indicated in the flowchart of 3.1.2. The first example uses a direct FE solver, and the second one uses an iterative solver due to its larger size (approximately 1 million elements).

Ex.7 3D MBB beam with concentrated load

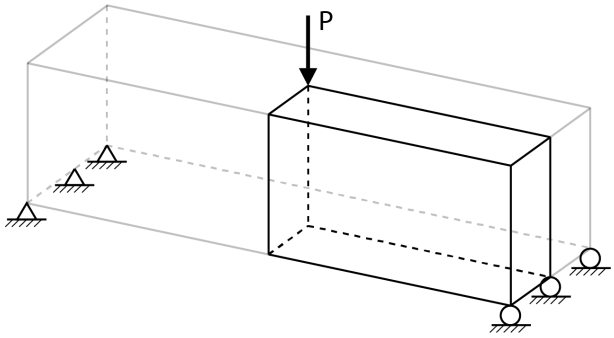
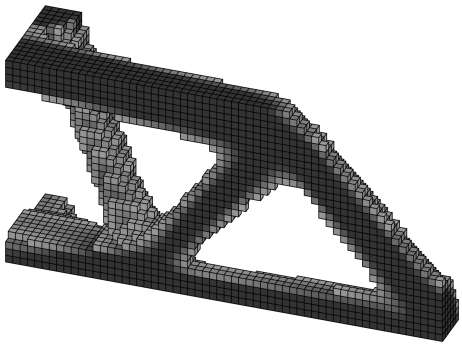


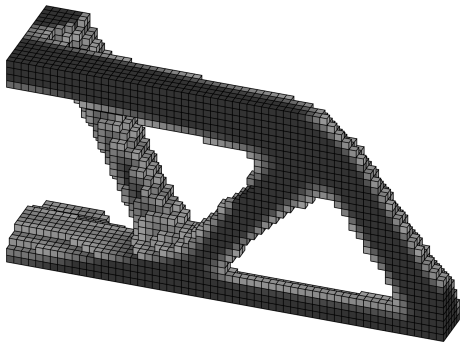
Figure 4.13: 3D simply supported beam with a concentrated load

This example is a 3D MBB beam with a concentrated load in the center of the top surface as shown in Figure 4.13. The actual computation model is the highlighted quarter due to the symmetry of the domain and boundary conditions.
The parameters are shown on the next page.

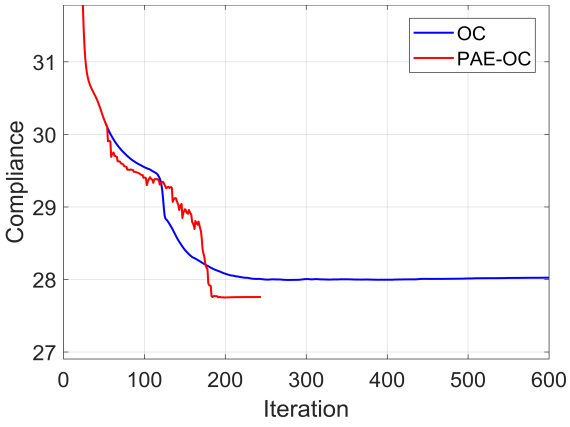
- Mesh $60 \times 30 \times 15$, volume fraction $V = 0.175$, penalization $p = 3$, filter radius $r = 4$, maximum iteration = 600
- AAR parameters: $m = 4$, $q = 4$, $\alpha = 0.9$, $\beta = 3 + loop/50 \leq 9$, $s = 50$



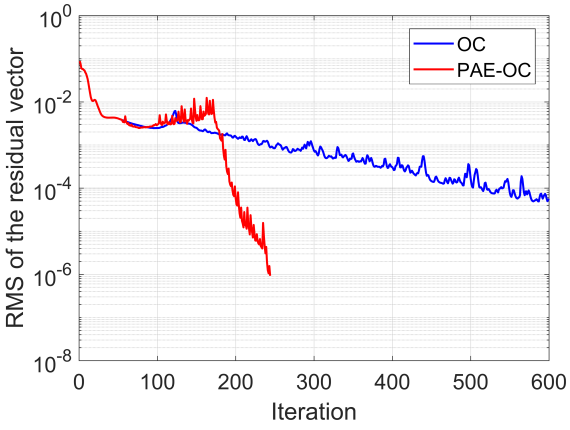
(a) Final design by standard OC (0.5 cut-off)



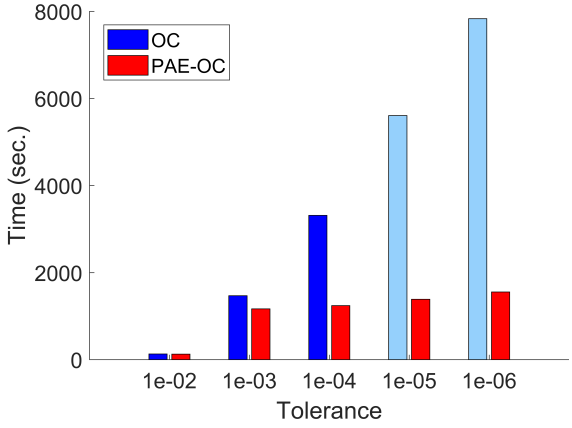
(b) Final design by PAE-OC (0.5 cut-off)



(c) Objective function history



(d) Residual history



(e) Consumed time (lighter blue indicates estimated value)

Figure 4.14: Ex.7 Design Results and Performance of standard OC and PAE-OC

Figure 4.14 shows the results of this example. The designs are displayed using a density cutoff of 0.5. Since the residual of the standard OC did not reach 10^{-5} within 600 iterations, the corresponding time matching 10^{-5} and 10^{-6} are obtained by linear extrapolation of the

blue residual curve. The 2 designs show some minor difference, for example, the sizes of the triangular holes. In terms of convergence speed, the PAE-OC beat the standard OC with a faster decline in both the objective and the residual.

Ex.8 3D cantilever beam with concentrated load

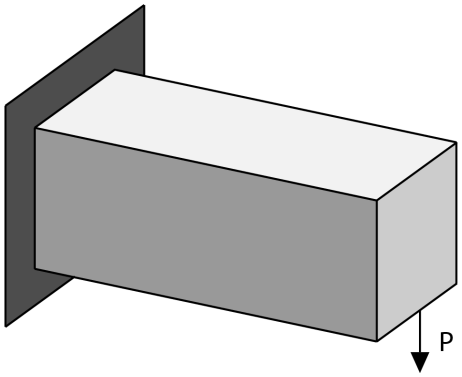
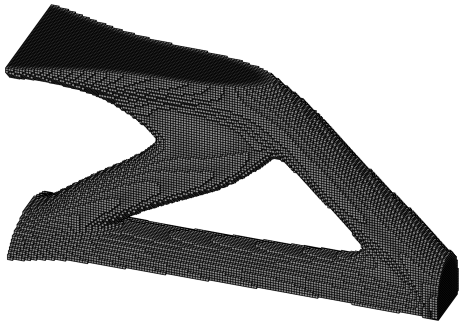


Figure 4.15: 3D cantilever beam with a concentrated load

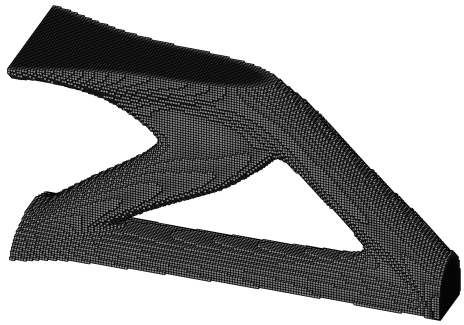
The second 3D example is a cantilever beam with a concentrated load at the lower edge of the tip as shown in Figure 4.15. The parameters are given below.

- Mesh $160 \times 80 \times 80$, volume fraction $V = 0.12$, penalization $p = 3$, filter radius $r = 6$, maximum iteration = 400
- AAR parameters: $m = 4$, $q = 4$, $\alpha = 0.9$, $\beta = 3 + loop/50 \leq 5$, $s = 25$

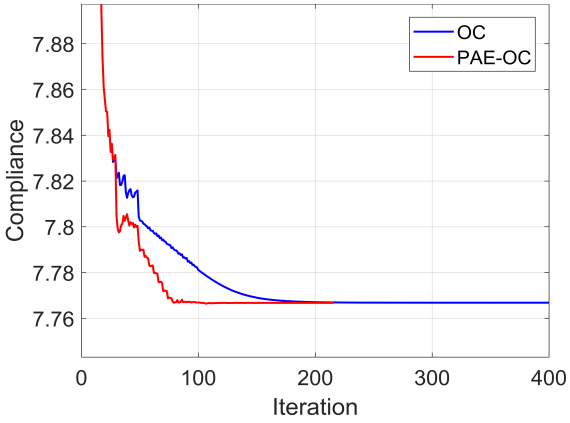
This example has $1,024,000$ elements and more than 3 million degrees of freedom. Linear systems of this size require iterative solvers. To solve this problem, we adopt Matlab's preconditioned conjugate gradient solver with the zero fill-in, incomplete Cholesky decomposition as the preconditioner (PCG IC(0)). To further save time in the linear system, we use a continuation strategy for the tolerance which decreases by 1 order of magnitude every 50 design cycles from 10^{-4} to 10^{-8}. Although this leads to larger displacement errors in the early stage, the design sensitivity is insensitive to the linear system's accuracy [6].



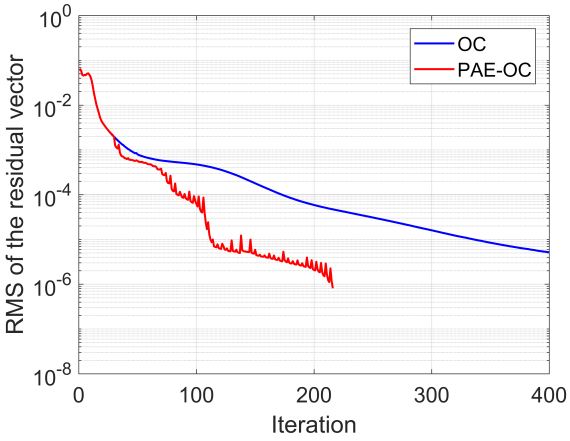
(a) Final design by standard OC (0.5 cut-off)



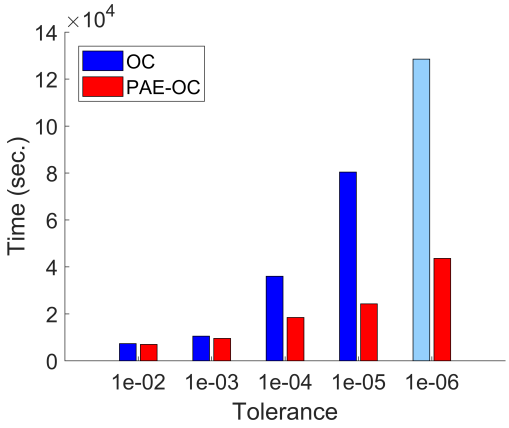
(b) Final design by PAE-OC (0.5 cut-off)



(c) Objective function history



(d) Residual history



(e) Consumed time (lighter blue indicates estimated value)

Figure 4.16: Ex.8 Design Results and Performance of standard OC and PAE-OC

Figure 4.16 shows the results of this example. Since the residual of the standard OC did not drop to 10^{-6} in 400 steps, the corresponding time (in light blue) is based on the linear extrapolation of the residual curve. As can be seen, PAE-OC converges faster than standard OC in both the objective and residual. This example shows that PAE-OC's performance is

insensitive to the non-exact iterative solver and the growth of the problems size, which makes room for PAE-OC's large-scale application.

4.4 Discussions

This chapter demonstrates PAE-OC's speedup in convergence from 4 perspectives: the objective, the residual, the computational time, and the design evolution. Via several examples, it is shown that PAE-OC can effectively reduce the number of iterations and computational time with negligible increase in the step-wise cost. Moreover, PAE-OC is robust subjected to the change of problem sizes, parameters, and the use of iterative solvers.

PAE's speedup mostly comes from the Anderson mixing part. From a heuristic perspective, the Anderson mixing accelerates the change of medium densities by extrapolation based on the information of previous steps.

Regarding the objective function, although the PAE-OC achieves objectives no higher than the standard OC in the presented examples, finding a better local optimum is not guaranteed. This is due to the non-convex nature of the topology optimization problem. Indeed, in some tests, standard OC converged to a design with a lower compliance, only with many more steps. In addition, PAE-OC is very probable to end up in a design with different topology from that generated by the standard OC if the design has many details.

Finally, we also note that PAE-OC's speedup in convergence can weaken if a very small filter radius is applied, although in most such cases, it still converges faster than the OC.

CHAPTER 5
CONCLUSIONS

Topology optimization with first-order updates suffers from slow convergence. In many cases, the optimization consumes a large number of design cycles to result in only a slight improvement in the performance and a small change in the design. To address the slow convergence, we propose to use fixed-point iteration methods to accelerate the first-order updates, which has not been studied previously in the topology optimization field.

Based on theoretical analyses and numerical tests, we analyze the pros and cons of several fixed-point iteration methods in the context of topology optimization. Simple mixing preserves the volume and the density bounds and stabilizes the design evolution, but it fails to effectively accelerate the convergence. Broyden's method does not preserve the volume and bounds in general, which can lead to instability and even divergence. It also requires a high computational cost if the approximated inverse Jacobian is updated and stored in matrix form. Anderson mixing preserves the volume and may violate the bounds slightly, which results in the better stability than the Broyden's method. It can effectively reduce the number of iterations provided the parameters are chosen within certain ranges. In addition, Anderson mixing has a lower step-wise cost than the Broyden's method. The PAE method, which combines simple mixing and Anderson mixing, performs the best in terms of stability, convergence speedup, and step-wise cost. Based on numerical studies, we discussed the influence of PAE's parameters on the convergence and summarize the optimal parameter ranges. Finally, we present the implementation details of the PAE-OC update.

With several 2D and 3D benchmarks, we demonstrate that the PAE-OC scheme can effectively reduce the number of iterations and computational time. The scheme is also more efficient in removing the slowly-varying intermediate densities than the OC. Moreover, PAE-OC's performance is insensitive to the change of problem sizes and boundary

conditions. Last but not least, we show that PAE-OC is scalable through a 3D example with more than 1 million elements.

For future studies, it is valuable to develop effective preconditioners tailored for the PAE-OC method or other fixed-point accelerated updates. This can further accelerate the convergence and save computation costs. Another direction is incorporating fixed-point iteration techniques with the MMA, which enables solving problems other than the compliance minimization.

REFERENCES

[1] O. Sigmund and K. Maute, "Topology optimization approaches," *Structural and Multidisciplinary Optimization*, vol. 48, no. 6, pp. 1031–1055, 2013.

[2] G. I. N. Rozvany, "A critical review of established methods of structural topology optimization," *Structural and Multidisciplinary Optimization*, vol. 37, no. 3, pp. 217–237, 2009.

[3] K. Svanberg, "The method of moving asymptotes - a new method for structural optimization," *International Journal for Numerical Methods in Engineering*, vol. 24, no. 2, pp. 359–373, 1987.

[4] P. Christensen and A. Klarbring, *An Introduction to Structural Optimization*. Springer Netherlands, 2009.

[5] M. P. Bendsoe and O. Sigmund, *Topology Optimization: Theory, Methods, and Applications*. Springer-Verlag Berlin Heidelberg, 2004.

[6] O. Amir and O. Sigmund, "On reducing computational effort in topology optimization: How far can we go?" *Structural and Multidisciplinary Optimization*, vol. 44, no. 1, pp. 25–29, 2011.

[7] H Smaoui, C. Fleury, and L. A. Schmit, "Advances in dual algorithms and convex approximation methods," in *29th Structures, Structural Dynamics and Materials Conference*, California Univ., Jul. 1988, p. 9.

[8] C. Fleury, "First and second order convex approximation strategies in structural optimization," *Structural optimization*, vol. 1, no. 1, pp. 3–10, 1989.

[9] P. Duisinx, W. Zhang, C. Fleury, V. H. Nguyen, and S. Haubruge, "A new separable approximation scheme for topological problems and optimization problems characterized by a large number of design variables," *Proceedings of the First World Congress of Structural and Multidisciplinary Optimization (WCSMO1)*, pp. 1 –8, 1995.

[10] M. Bruyneel, P. Duysinx, and C. Fleury, "A family of MMA approximations for structural optimization," *Structural and Multidisciplinary Optimization*, vol. 24, no. 4, pp. 263–276, 2002.

[11] S. Rojas-Labanda and M. Stolpe, "An efficient second-order SQP method for structural topology optimization," *Structural and Multidisciplinary Optimization*, vol. 53, no. 6, pp. 1315–1333, 2016.

[12] S. Rojas-Labanda and M. Stolpe, "Solving large-scale structural topology optimization problems using a second-order interior point method."

[13] S. Wang, E. d. Sturler, and G. H. Paulino, "Large-scale topology optimization using preconditioned Krylov subspace methods with recycling," *International Journal for Numerical Methods in Engineering*, vol. 69, no. 12, pp. 2441–2468, 2007.

[14] O. Amir, N. Aage, and B. S. Lazarov, "On multigrid-CG for efficient topology optimization," *Structural and Multidisciplinary Optimization*, vol. 49, no. 5, pp. 815–829, 2014.

[15] O. Amir, M. Stolpe, and O. Sigmund, "Efficient use of iterative solvers in nested topology optimization," *Structural and Multidisciplinary Optimization*, vol. 42, no. 1, pp. 55–72, 2010.

[16] N. Aage, E. Andreassen, and B. S. Lazarov, "Topology optimization using PETSc: An easy-to-use, fully parallel, open source topology optimization framework," *Structural and Multidisciplinary Optimization*, vol. 51, no. 3, pp. 565–572, 2015.

[17] N. Aage, E. Andreassen, B. S. Lazarov, and O. Sigmund, "Giga-voxel computational morphogenesis for structural design," *Nature*, vol. 550, 2017.

[18] O. Amir, M. P. Bendse, and O. Sigmund, "Approximate reanalysis in topology optimization," *International Journal for Numerical Methods in Engineering*, vol. 78, no. 12, pp. 1474–1491, 2009.

[19] P. P. Pratapa, P. Suryanarayana, and J. E. Pask, "Anderson acceleration of the Jacobi iterative method: An efficient alternative to Krylov methods for large, sparse linear systems," *Journal of Computational Physics*, vol. 306, pp. 43 –54, 2016.

[20] P. Suryanarayana, P. P. Pratapa, and J. E. Pask, "Alternating Anderson -Richardson method: An efficient alternative to preconditioned Krylov methods for large, sparse linear systems," arXiv.org, Tech. Rep. arXiv:1606.08740, 2018.

[21] E. Andreassen, A. Clausen, M. Schevenels, B. S. Lazarov, and O. Sigmund, "Efficient topology optimization in matlab using 88 lines of code," *Structural and Multidisciplinary Optimization*, vol. 43, no. 1, pp. 1–16, 2011.

[22] K. Svanberg, "A class of globally convergent optimization methods based on conservative convex separable approximations," vol. 12, no. 2, pp. 555–573, 2002.

[23] M. Bruyneel and C. Fleury, "Composite structures optimization using sequential convex programming," *Advances in Engineering Software*, vol. 33, no. 7, pp. 697 –711, 2002, Engineering Computational Technology & Computational Structures Technology.

[24] P. Duisinx, "Solution of topology optimisation problems with sequential convex programming - structural approximations," Topology Optimization Theory, Methods, and Applications DCAMM, Copenhagen, Jul. 2017.

[25] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical Programming*, vol. 91, no. 2, pp. 201–213, 2002.

[26] P. Gill, W. Murray, and M. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Review*, vol. 47, no. 1, pp. 99–131, 2005.

[27] S. Rojas-Labanda and M. Stolpe, "Benchmarking optimization solvers for structural topology optimization," *Structural and Multidisciplinary Optimization*, vol. 52, no. 3, pp. 527–547, 2015.

[28] Mathworks, *Optimization toolbox users guide r 2013 b*, 2013.

[29] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[30] C. G. Broyden, "A class of methods for solving nonlinear simultaneous equations," *Math. Comp.*, vol. 19, pp. 577 –593, 1965.

[31] D. G. Anderson, "Iterative procedures for nonlinear integral equations," *Journal of the Association for Computing Machinery*, vol. 12, no. 4, pp. 547 –560, 1965.

[32] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*. Society for Industrial and Applied Mathematics, 1995.

[33] V. Eyert, "A comparative study on methods for convergence acceleration of iterative vector sequences," *Journal of Computational Physics*, vol. 124, no. 2, pp. 271 –285, 1996.

[34] H. Fang and Y. Saad, "Two classes of multisecant methods for nonlinear acceleration," *Numerical Linear Algebra with Applications*, vol. 16, no. 3, pp. 197–221, 2009.

[35] J. Dennis and J. J. Moré, "Quasi-Newton methods, motivation and theory," *SIAM Rev.*, vol. 19, no. 1, pp. 46 –89, 1977.

[36] A. Toth and C. Kelley, "Convergence analysis for Anderson acceleration," *SIAM J. Numer. Anal.*, vol. 53, no. 2, pp. 805 –819, 2015.

[37] H. F. Walker and P. Ni, "Anderson acceleration for fixed-point iterations," *SIAM J. Numer. Anal.*, vol. 49, no. 4, pp. 1715 –1735, 2011.

[38] A. S. Banerjee, P. Suryanarayana, and J. E. Pask, "Periodic Pulay method for robust and efficient convergence acceleration of self-consistent field iterations," *Chemical Physics Letters*, vol. 647, pp. 31 –35, 2016.

[39] K. Liu and A. Tovar, "An efficient 3d topology optimization code written in matlab," *Structural and Multidisciplinary Optimization*, vol. 50, no. 6, pp. 1175–1196, 2014.